# 統計モデリング入門 2016 (f)

## 階層ベイズモデル Hierarchical Bayesian Model

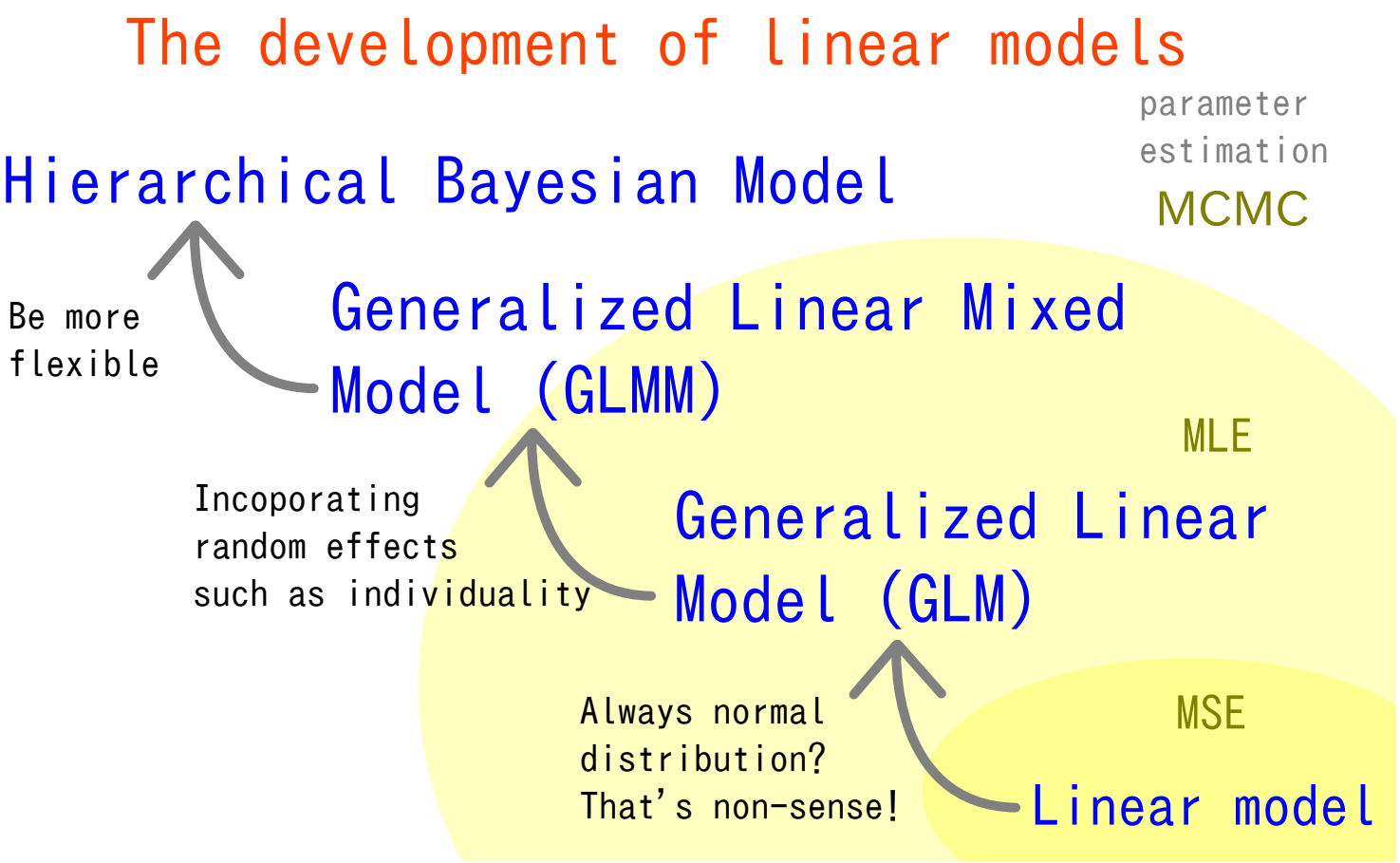
**久保拓弥** kubo@ees.hokudai.ac.jp

**北大環境科学院の講義** http://goo.gl/76c4i

2016–07–27

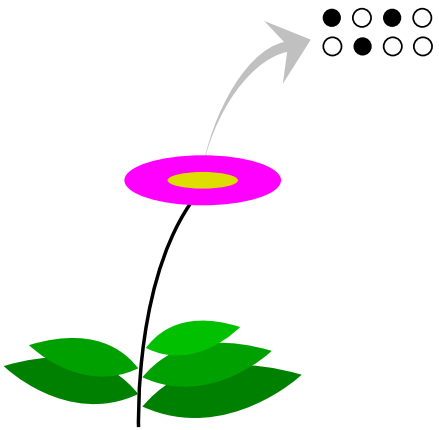
**ファイル更新時刻**: 2016–07–27 16:06

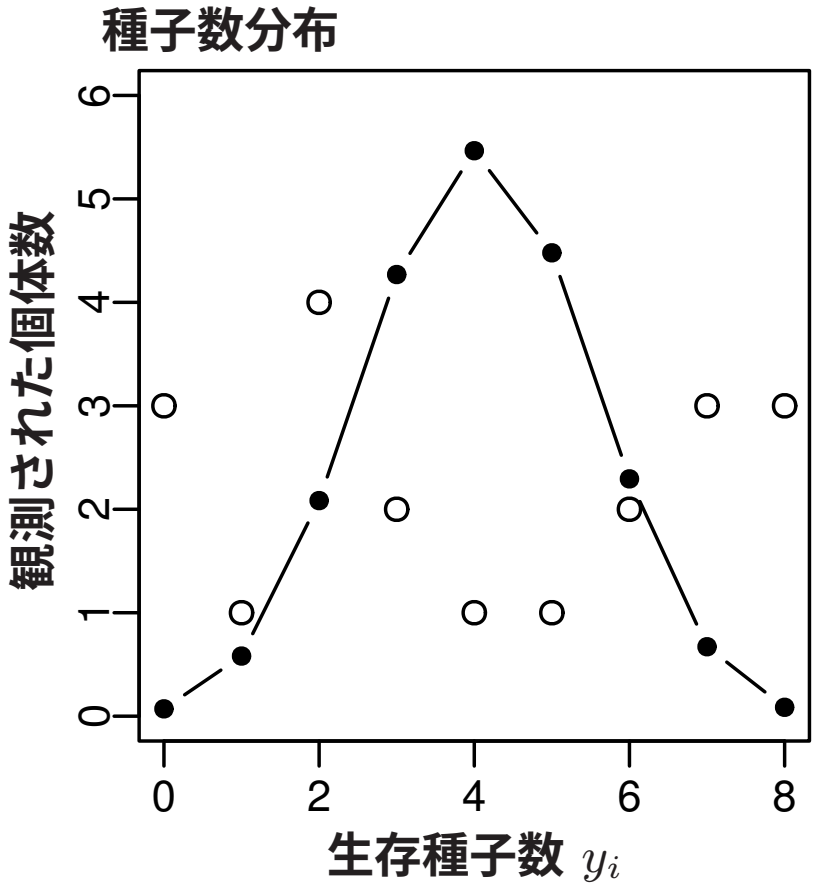
# 今日の統計モデル: 階層ベイズモデル




そして Markov Chain Monte Carlo (MCMC)
を使った Bayesian Estimation (ベイズ推定)

# Why? GLM is not enough ...






$N$ 個のうち $y$ 個
…という形式のデータ
なのに
二項分布ではまったく
説明できない?

**階層ベイズモデルが必要!**

Apply Hierarchical Bayesian Model (HBM)!

# 今日のハナシ

1. MCMC サンプリングのための example 例題
   logistic regression: binomial distribution
2. 同じような推定を MCMC でやってみる
   最尤推定と Markov chain Monte Carlo (MCMC) はちがう!
3. Softwares for MCMC sampling
   “Gibbs sampling” などが簡単にできるような……
4. GLMM と階層ベイズモデル
   GLMM のベイズモデル化
5. 階層ベイズモデルの estimation 推定
   ソフトウェア JAGS を使ってみる

# 1. MCMC サンプリングのための 例題

example

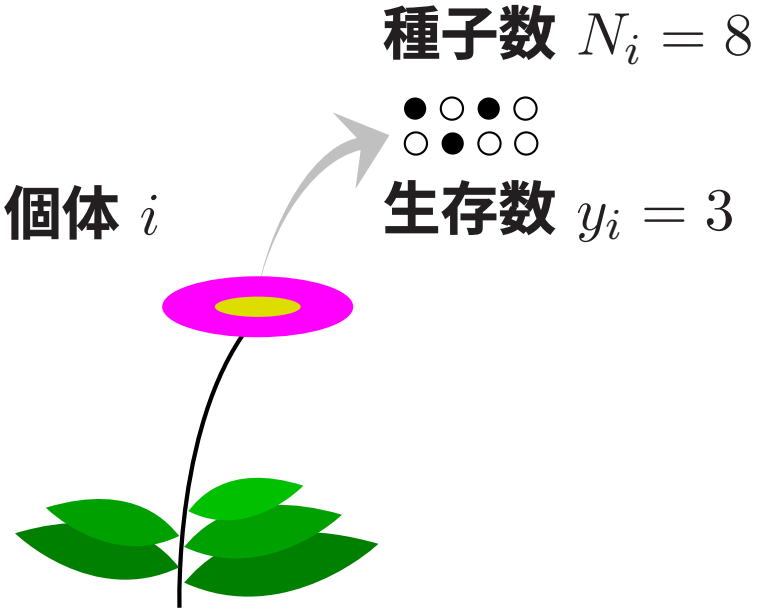
logistic regression: binomial distribution

and logit link function

example　seed survivorship, again

# 例題：植物の種子の生存確率

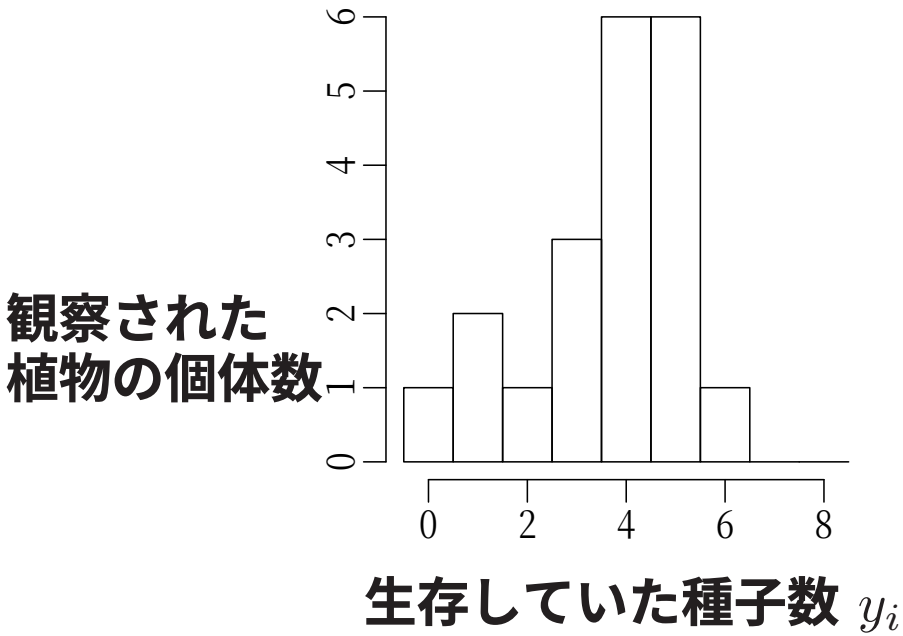
- 架空植物の種子の生存を調べた
- **種子**: 生きていれば発芽する
  - どの個体でも 8 個 の種子を調べた
- **生存確率**: ある種子が生きている確率




- データ: 植物 20 個体，合計 160 種子の生存の有無を調べた
- 73 種子が生きていた — このデータを統計モデル化したい

# たとえばこんなデータが得られたとしましょう

| 個体ごとの生存数 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 観察された個体数 | 1 | 2 | 1 | 3 | 6 | 6 | 1 | 0 | 0 |




これは個体差なしの均質な集団

# 生存確率 $q$ と　二項分布　の関係

binomial distribution

- 生存確率を推定するために**二項分布** という確率分布を使う
- 個体 $i$ の $N_i$ 種子中 $y_i$ 個が生存する確率

$$p(y_i \mid q) = \binom{N_i}{y_i} q^{y_i} (1-q)^{N_i - y_i},$$

- ここで仮定していること
  - **個体差はない**
  - つまり **すべての個体で同じ生存確率** $q$

# 尤度 (ゆうど): 20 個体ぶんのデータが観察される確率

- 観察データ $\{y_i\}$ が確定しているときに
- パラメータ $q$ は値が自由にとりうると考える

likelihood

- 尤度 は 20 個体ぶんのデータが得られる確率の積，パラメータ $q$ の関数として定義される

$$L(q|\{y_i\}) = \prod_{i=1}^{20} p(y_i \mid q)$$

| 個体ごとの生存数 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 観察された個体数 | 1 | 2 | 1 | 3 | 6 | 6 | 1 | 0 | 0 |

# 対数尤度方程式と最尤推定

- この尤度 $L(q \mid \text{データ})$ を最大化するパラメータの推定量 $\hat{q}$ を計算したい
- 尤度を対数尤度になおすと

$$\log L(q \mid \text{データ}) = \sum_{i=1}^{20} \log \binom{N_i}{y_i} + \sum_{i=1}^{20} \{y_i \log(q) + (N_i - y_i) \log(1-q)\}$$

- この対数尤度を最大化するように未知パラメーター $q$ の値を決めてやるのが**最尤推定**

maximum likelihood estimation

# 最尤推定 (MLE)　とは何か

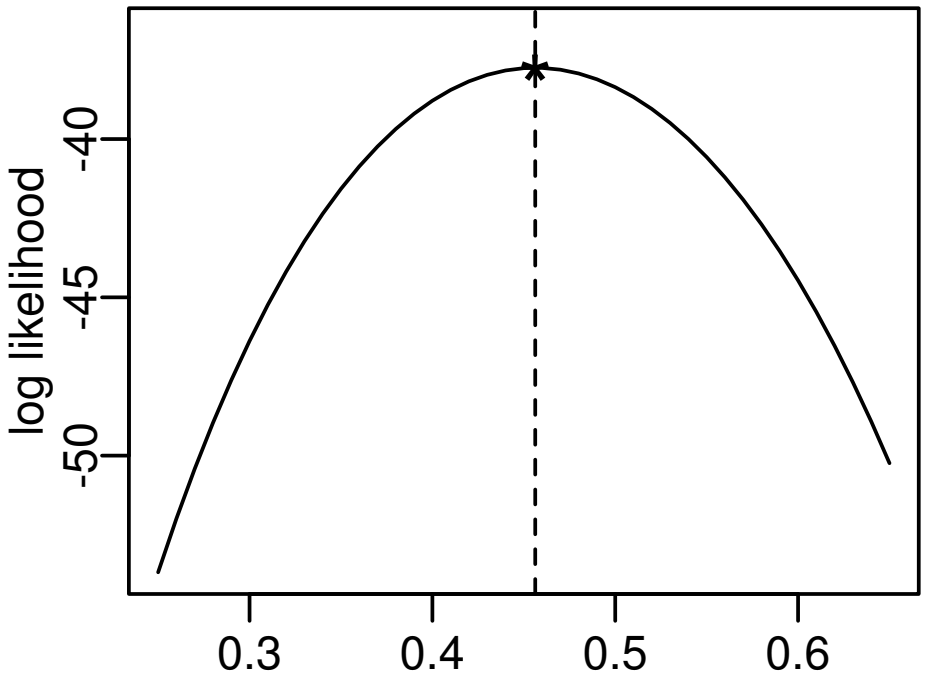
- **対数尤度 $L(q \mid$ データ$)$ が最大になるパラメーター $q$ の値をさがしだすこと**

- **対数尤度 $\log L(q \mid$ データ$)$ を $q$ で偏微分して 0 となる $\hat{q}$ が対数尤度最大**

$$\partial \log L(q \mid \text{データ})/\partial q = 0$$

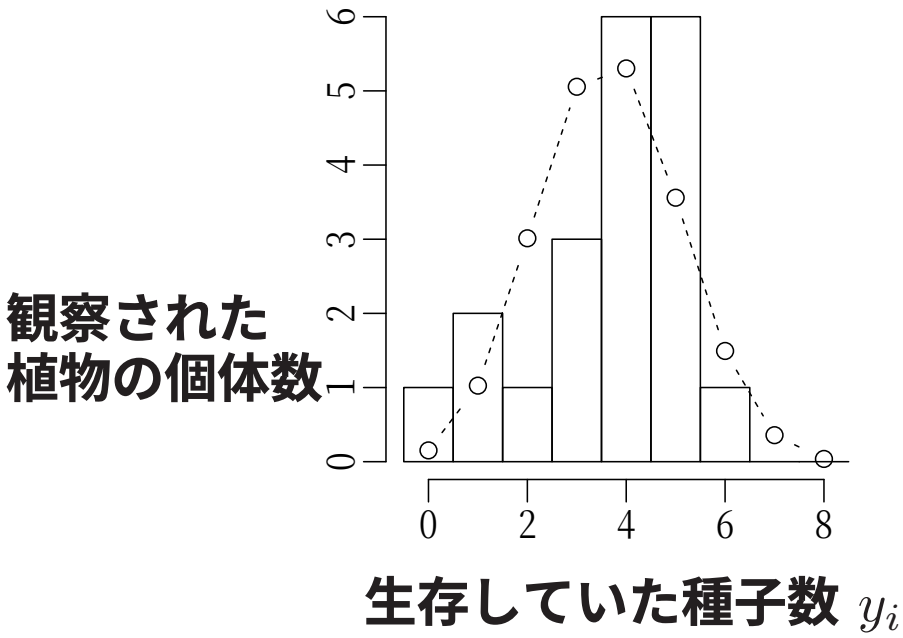
- **生存確率 $q$ が全個体共通の場合の最尤推定量・最尤推定値は**

$$\hat{q} = \frac{\text{生存種子数}}{\text{調査種子数}} = \frac{73}{160} = 0.456 \text{ ぐらい}$$



# 二項分布で説明できる 8 種子中 $y_i$ 個の生存

$\hat{q} = 0.46$ **なので** $\binom{8}{y} 0.46^y \ 0.54^{8-y}$



# 2. 同じような推定を MCMC でやってみる

## 最尤推定と Markov chain Monte Carlo (MCMC) はちがう!

### そして “なんとなーく” ベイズ統計モデルと関連づけ

# ここでやること: 尤度と MCMC の関係を考える

- さきほどの簡単な例題 (生存確率) のデータ解析を
- 最尤推定ではなく
- Markov chain Monte Carlo (MCMC) 法のひとつである**メトロポリス法** (Metropolis method) であつかう
- 得られる結果: 「パラメーターの値の分布」……??

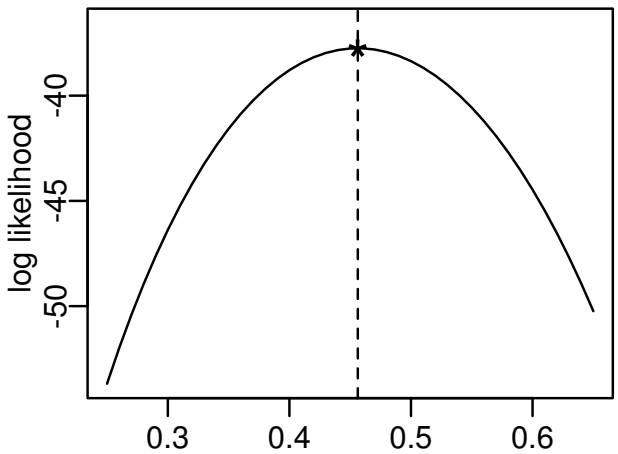
MCMC をもちださなくてもいい簡単すぎる問題

説明のためあえてメトロポリス法を適用してみる

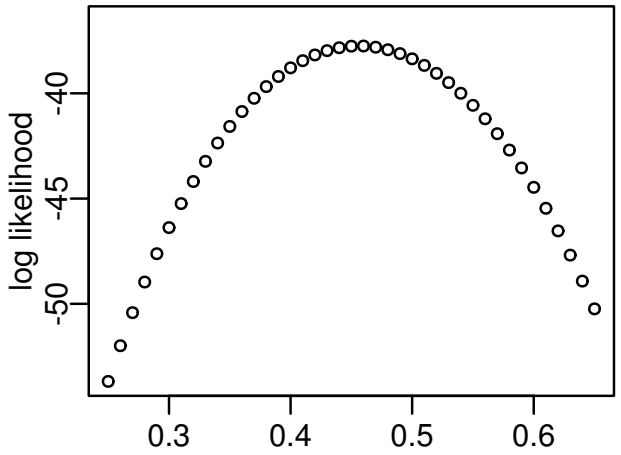
# メトロポリス法を説明するための準備

連続的な対数尤度関数

$\log L(q)$




離散化: $q$ がとびとびの値をとる



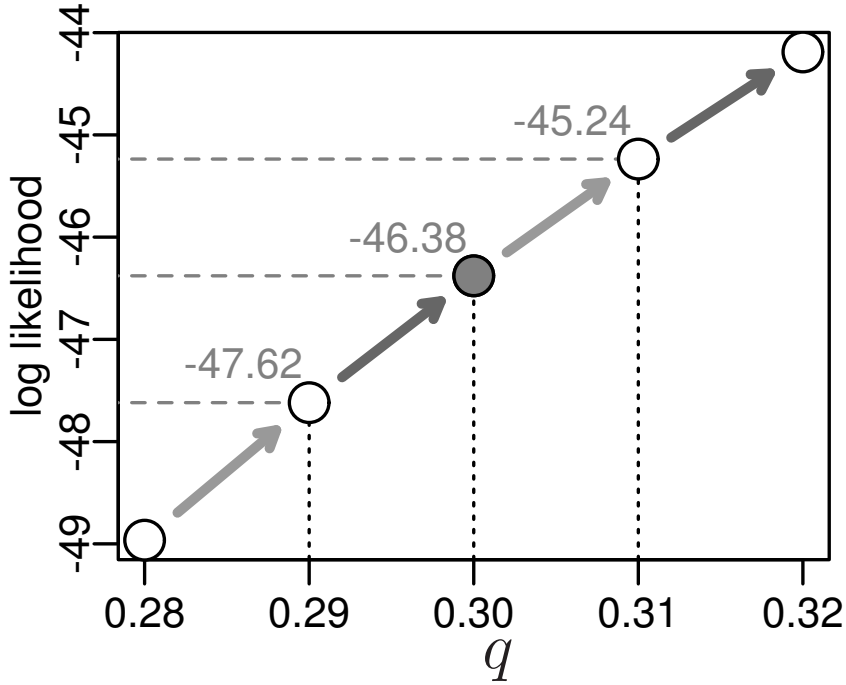

**説明を簡単にするため**

**生存確率 $q$ の軸を離散化する**

(実際には離散化する必要などない)

# 試行錯誤による $q$ の最尤推定値の探索

## ちょっと効率の悪い「試行錯誤の最尤推定」




1. $q$ の値の「行き先」を「両隣」どちらかにランダムに決める
2. 「行き先」が現在の尤度より高ければ，$q$ の値をそちらに変更
3. 尤度が変化しなくなるまで $(1), (2)$ をくりかえす

# メトロポリス法のルール: この例題の場合

1. **パラメーター $q$ の初期値を選ぶ**

   (ここでは $q$ の初期値が $0.3$)

2. **$q$ を増やすか減らすかをランダムに決める**

   (新しく選んだ $q$ の値を $q_{\text{new}}$ としましょう)

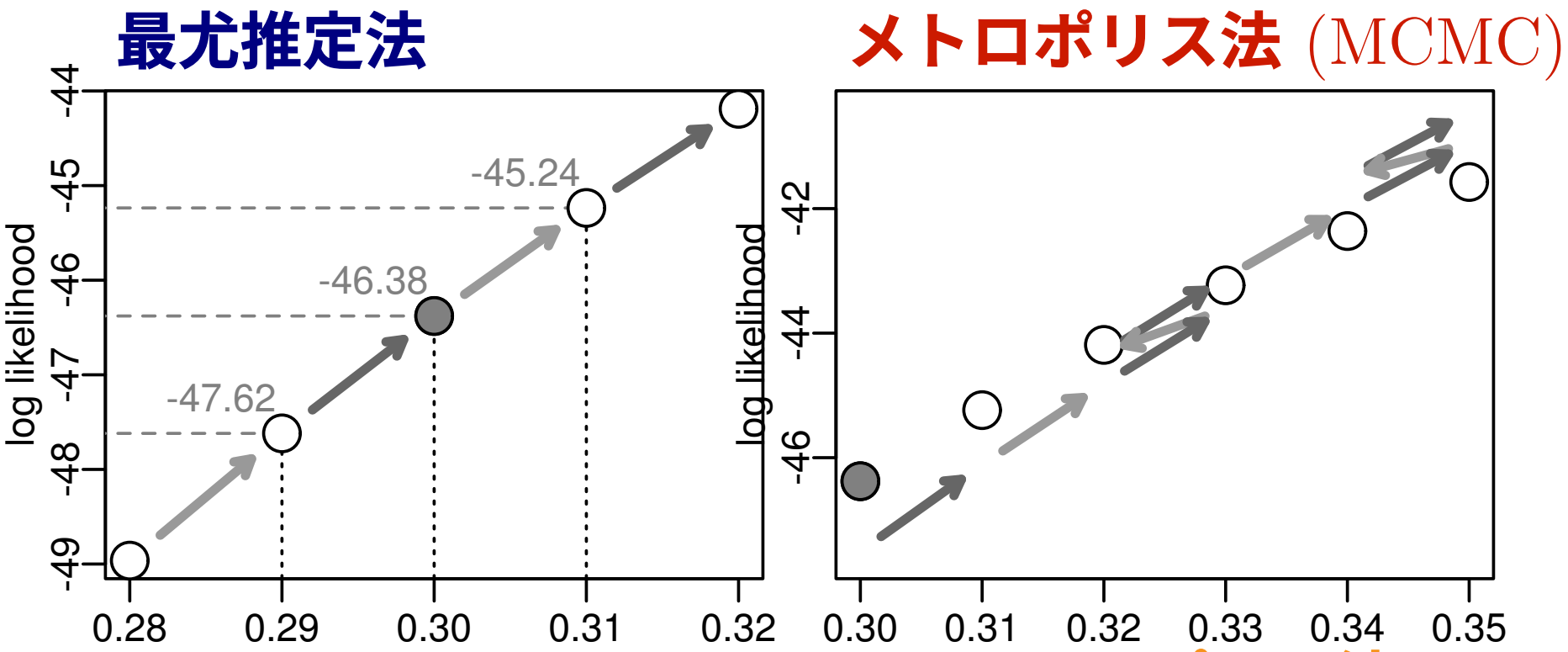
3. **$q_{\text{new}}$ における尤度 $L(q_{\text{new}})$ ともとの尤度 $L(q)$ を比較**

   - $L(q_{\text{new}}) \geq L(q)$ **(あてはまり改善)**: $q \leftarrow q_{\text{new}}$
   - $L(q_{\text{new}}) < L(q)$ **(あてはまり改悪)**:
     - **確率** $r = L(q_{\text{new}})/L(q)$ **で** $q \leftarrow q_{\text{new}}$
     - **確率** $1 - r$ **で** $q$ **を変更しない**

4. **手順 $2$. にもどる**

(**$q = 0.01$ や $q = 0.99$ でどうなるんだ，といった問題は省略**)

# メトロポリス法のルールで $q$ を動かす



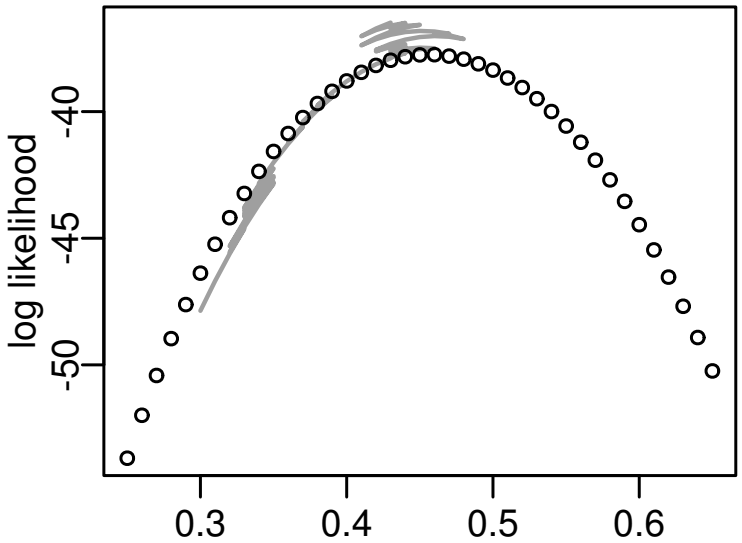

メトロポリス法だと

「単調な山のぼり」にはならない

# 対数尤度関数の「山」でうろうろする $q$ の値

**メトロポリス法 (そして一般の MCMC) は**

**最適化ではない**

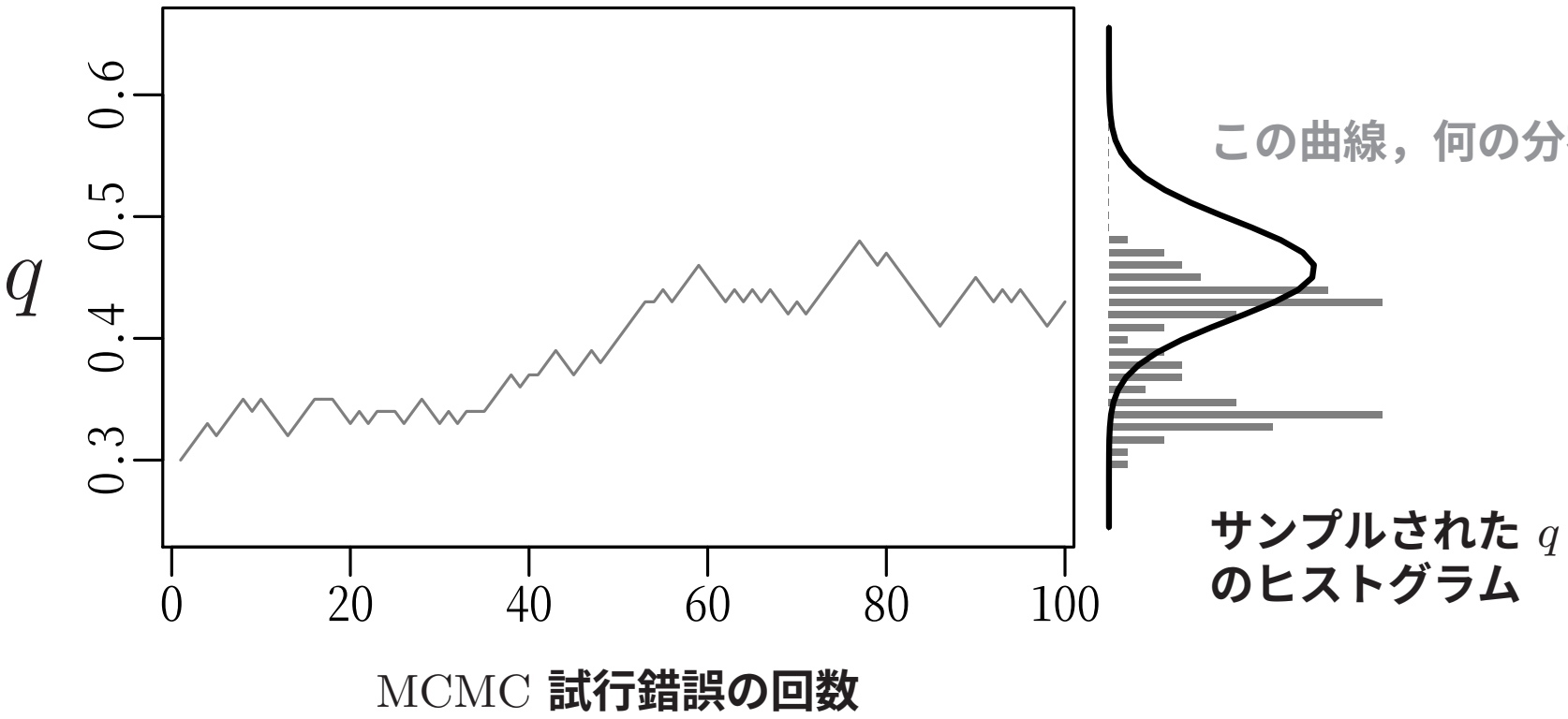



**ときどきはでに落っこちる**

**何のためにこんなことをやるのか?**

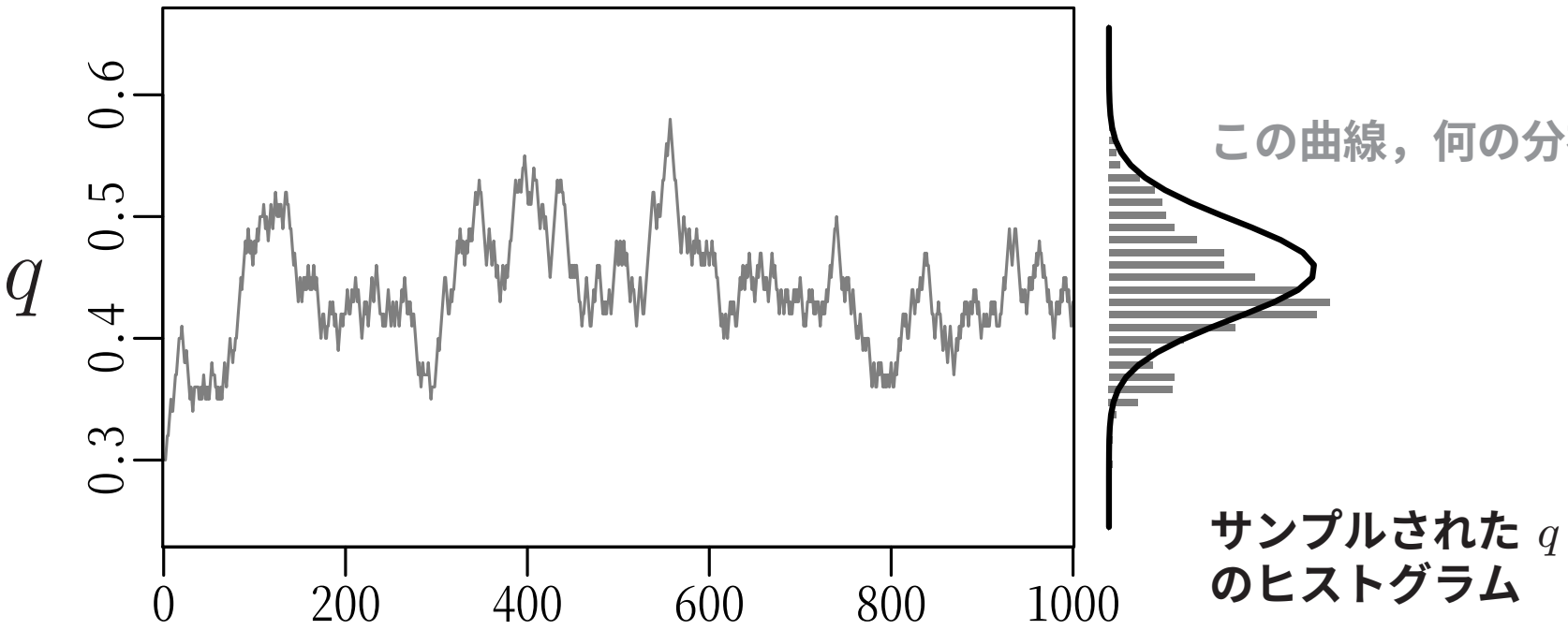
**$q$ の変化していく様子を記録してみよう**

# ステップごとに $q$ の値をサンプリング



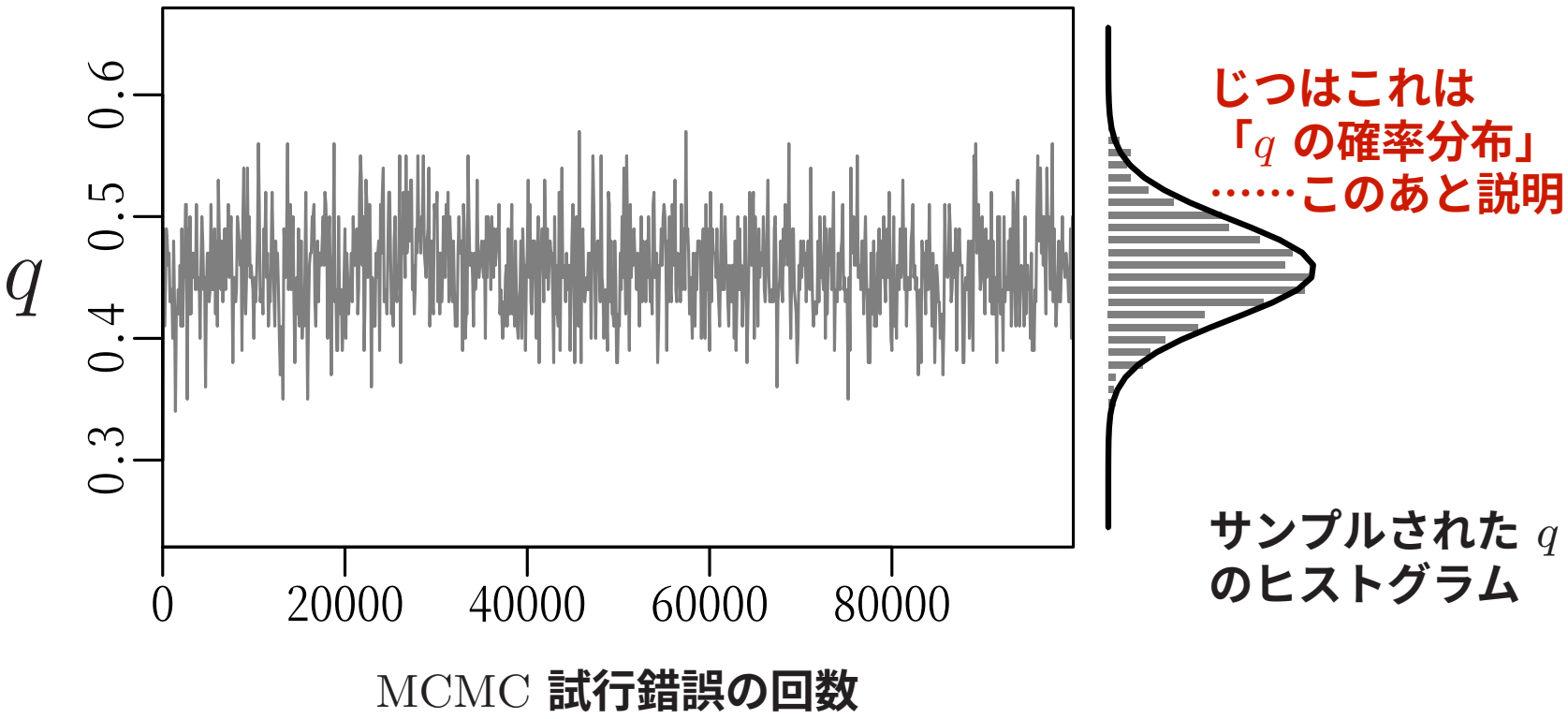

**もっと試行錯誤してみたほうがいいのかな?**

# もっと長くサンプリングしてみる




**まだまだ……?**

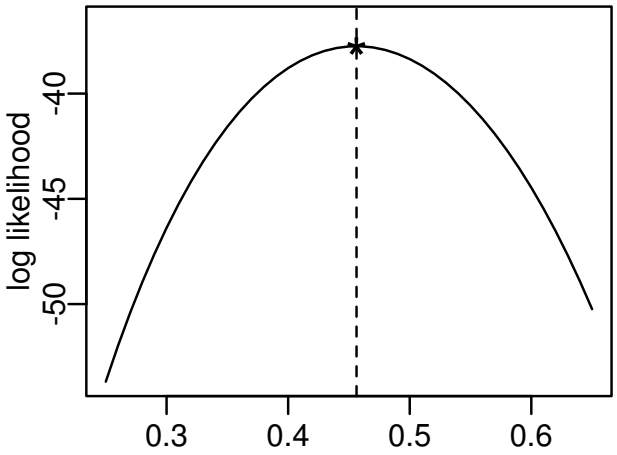
# もっともっと長くサンプリングしてみる



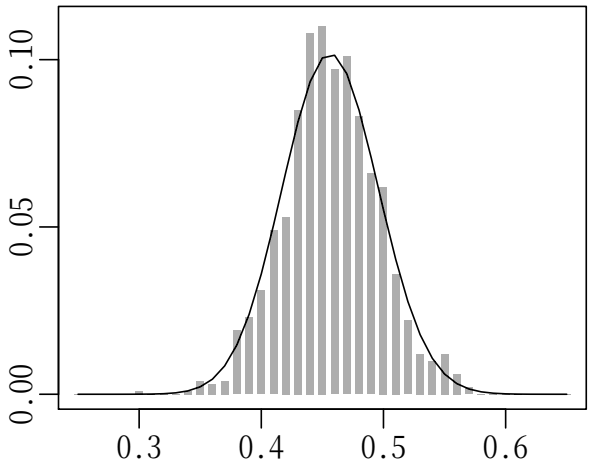

**なんだか，ある「山」のかたちにまとまったぞ?**

# MCMC は何をサンプリングしている?

**対数尤度** $\log L(q)$




**尤度** $L(q)$ **に比例する確率分布**



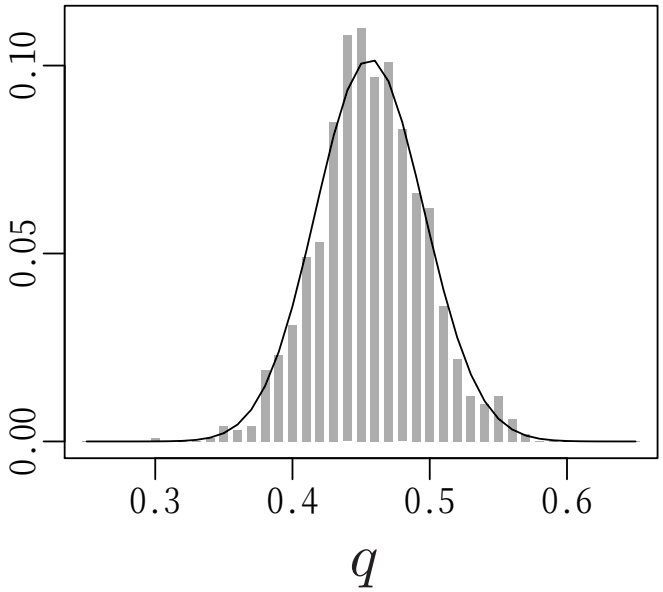

**尤度に比例する確率分布からのランダムサンプル**

**最尤推定はパラメーターの値の点推定**

MCMC **は "パラメーターの事後分布" (←推定したいこと)**

**は こういう分布ですよ と推定している**

# MCMC の結果として得られた $q$ の経験分布

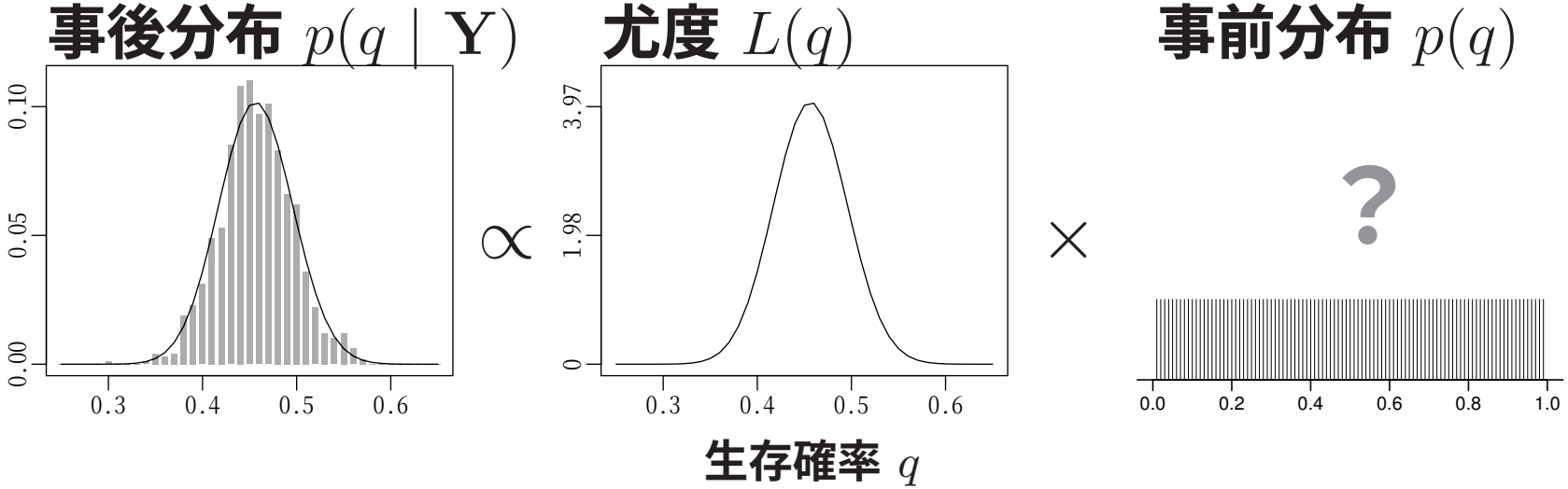



- データと統計モデル (二項分布) を決めて，MCMC サンプリングすると，$p(q)$ からのランダムサンプルが得られる
- このランダムサンプルをもとに，$q$ の平均や 95% 区間などがわかる— 便利じゃないか!

# ベイズ統計モデルの推定

**統計モデルとデータにもとづいて事後分布の推定**

- パラメーター数の少ないベイズモデルであれば，尤度の数値計算やメトロポリス法で可能
- パラメーター数の多い複雑な統計モデルであれば，**あとで説明する** サンプリングソフトウェアを使用する



# 3. Softwares for MCMC sampling

## “Gibbs sampling” などが簡単にできるような……

**事後分布から効率よくサンプリングしたい**

# 統計ソフトウェア R

http://www.r-project.org/

# 簡単な GLMM なら R だけで推定可能

- R にはいろいろな GLMM の最尤推定関数が準備されている ……
  - `library(glmmML)` の `glmmML()`
  - `library(lme4)` の `lmer()`
  - `library(nlme)` の `nlme()` (正規分布のみ)
- しかし もうちょっと複雑な GLMM，たとえば個体差 + 地域差をいれた統計モデルの最尤推定は かなり難しい (ヘンな結果が得られたりする)
- 積分がたくさん入っている尤度関数の評価がしんどい

# どのようなソフトウェアで MCMC 計算するか?

1. **自作プログラム**
   - **利点**: 問題にあわせて自由に設計できる
   - **欠点**: 階層ベイズモデル用の MCMC プログラミング，けっこうめんどう
2. R のベイズな package
   - **利点**: 空間ベイズ統計など便利な専用 package がある
   - **欠点**: 汎用性，とぼしい
3. **“BUGS” で “Gibbs sampler” なソフトウェア**
   - **利点**: 幅ひろい問題に適用できて，便利
   - **欠点というほどでもないけど，多少の勉強が必要**
   - **えーっと** “Gibbs sampler” **って何**?

# さまざまな MCMC アルゴリズム

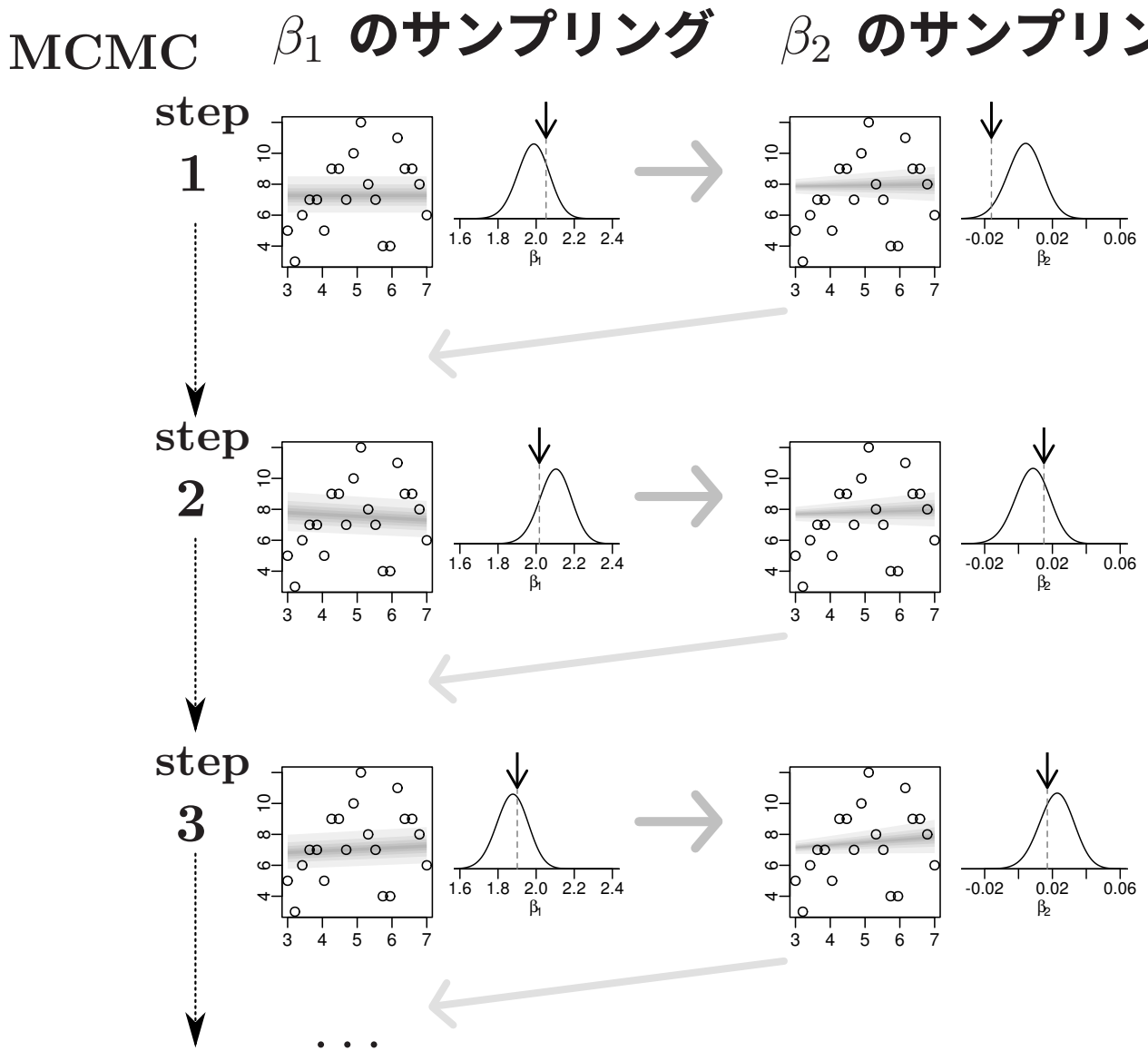
## いろいろな MCMC

- **メトロポリス法**: 試行錯誤で値を変化させていく MCMC
  - メトロポリス・ヘイスティングス法: その改良版
- **ギブス・サンプリング**: 条件つき確率分布を使った MCMC
  - 複数の変数 (パラメーター・状態) を効率よくサンプリング

# Gibbs sampling とは何か?

- MCMC アルゴリズムのひとつ
- 複数のパラメーターの MCMC サンプリングに使う
- 例: パラメーター $\beta_1$ と $\beta_2$ の Gibbs sampling
  1. $\beta_2$ に何か適当な値を与える
  2. $\beta_2$ の値はそのままにして，その条件のもとでの $\beta_1$ の MCMC sampling をする (条件つき事後分布)
  3. $\beta_1$ の値はそのままにして，その条件のもとでの $\beta_2$ の MCMC sampling をする (条件つき事後分布)
  4. 2. – 3. をくりかえす
- 教科書の第 9 章の例題で説明

# 図解: Gibbs sampling (統計モデリング入門の第 9 章)

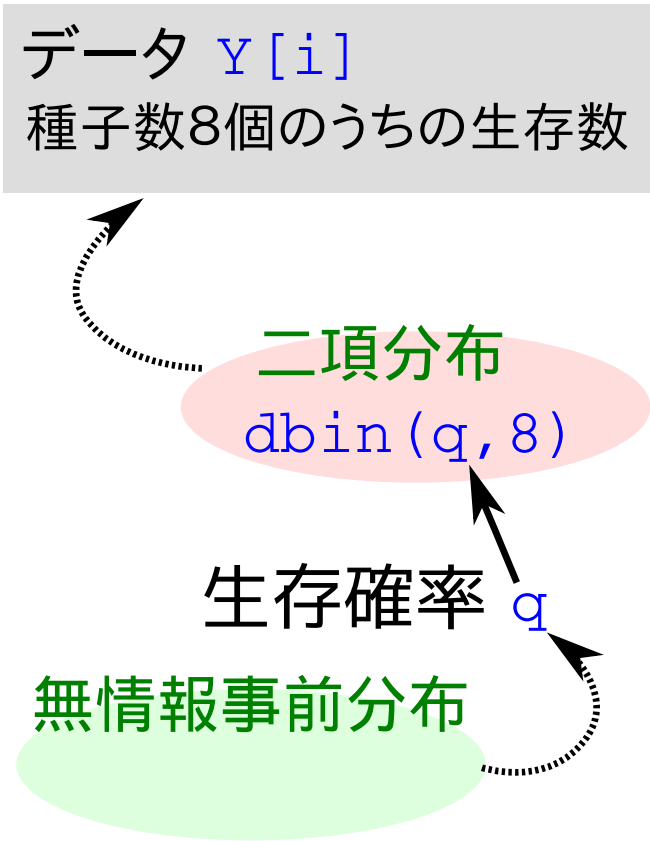


# 便利な "BUGS" 汎用 Gibbs sampler たち

- BUGS 言語 (+ っぽいもの) でベイズモデルを記述できるソフトウェア
  - WinBUGS — 歴史を変えて……さようなら?
  - OpenBUGS — 予算が足りなくて停滞?
  - **JAGS** — お手軽で良い，どんな OS でも動く
  - Stan — いま一番の注目
    — 今日は紹介しませんが ……
- リンク集: `http://hosho.ees.hokudai.ac.jp/~kubo/ce/BayesianMcmc.html`

えーと……BUGS 言語って何?

# このベイズモデルを BUGS 言語で記述したい




## BUGS 言語コード

```
for (i in 1:N.sample) {
  Y[i] ~ dbin(q, 8)
}
q ~ dunif(0.0, 1.0)
```

**矢印は手順ではなく，依存関係をあらわしている**

BUGS **言語**: **ベイズモデルを記述する言語**

Spiegelhalter et al. 1995. BUGS: Bayesian Using Gibbs Sampling version 0.50.

# いろいろな OS で使える JAGS3.4.x

- R core team のひとり Martyn Plummer さんが開発
  - Just Another Gibbs Sampler
- C++ で実装されている
  - R がインストールされていることが必要
- Linux, Windows, Mac OS X バイナリ版もある
- 開発進行中
- R からの使う: `library(rjags)`

# JAGS を R の “したうけ” として使う

モデルの構造

データと
パラメーター
の初期値

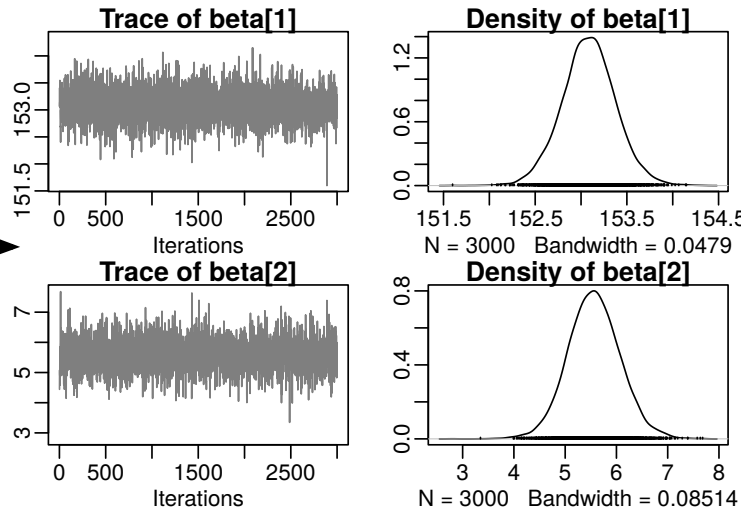
サンプリング
の詳細

Input

BUGS言語

JAGS

事後分布からの
ランダムサンプル




Output

# R から JAGS にこんなかんじで仕事を命じる (1 / 3)

```
library(rjags)
library(R2WinBUGS) # to use write.model()

model.bugs <- function()
{
  for (i in 1:N.data) {
    Y[i] ~ dbin(q, 8) # 二項分布にしたがう
  }
  q ~ dunif(0.0, 1.0) # q の事前分布は一様分布
}
file.model <- "model.bug.txt"
write.model(model.bugs, file.model) # ファイル出力

# 次につづく……
```

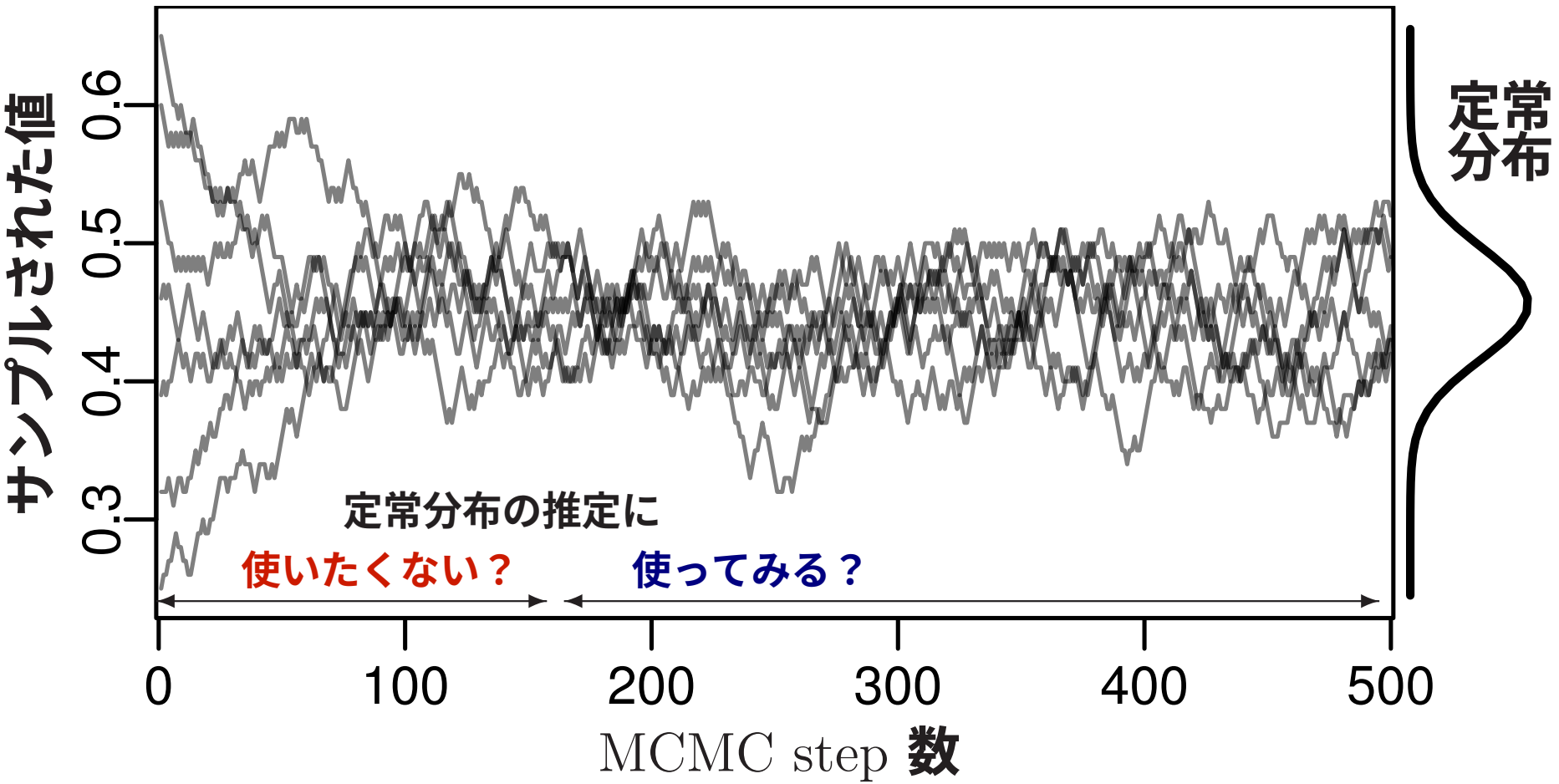
# R から JAGS にこんなかんじで仕事を命じる (2 / 3)

```
load("mcmc.RData") # (data.RData ではなく mcmc.RData!!)
list.data <- list(Y = data, N.data = length(data))
inits <- list(q = 0.5)
n.burnin <- 1000
n.chain <- 3
n.thin <- 1
n.iter <- n.thin * 1000

model <- jags.model(
  file = file.model, data = list.data,
  inits = inits, n.chain = n.chain
)

# まだ次につづく……
```

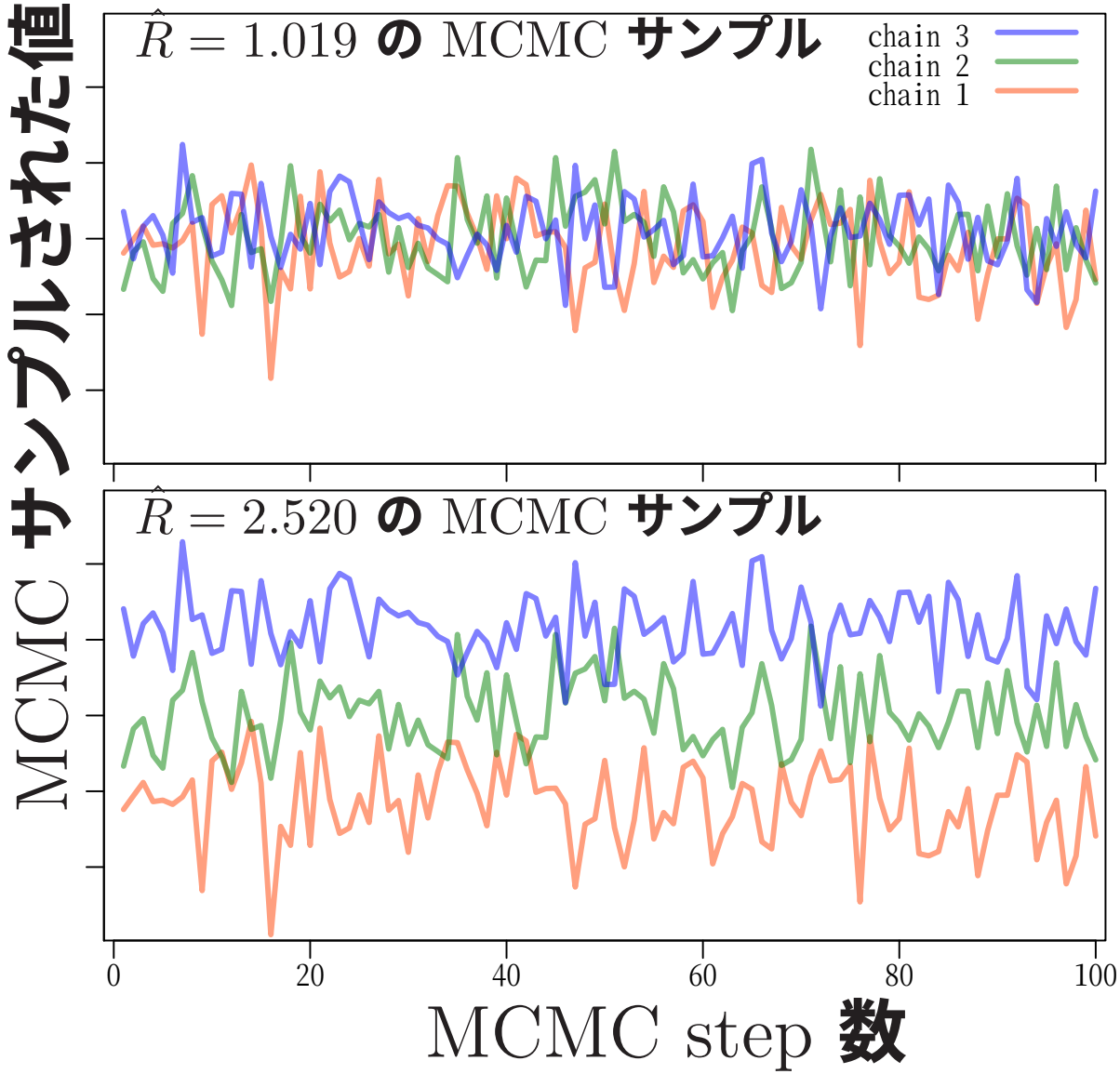
# R から JAGS にこんなかんじで仕事を命じる (3 / 3)

```
# burn-in
update(model, n.burnin) # burn in

# サンプリング結果を post.mcmc.list に格納
post.mcmc.list <- coda.samples(
  model = model,
  variable.names = names(inits),
  n.iter = n.iter,
  thin = n.thin
)
#  おわり
```

# burn in って何? → 「使いたくない」長さの指定



# 試行間で差がないかを「診断」する




まあ，
いいかな……

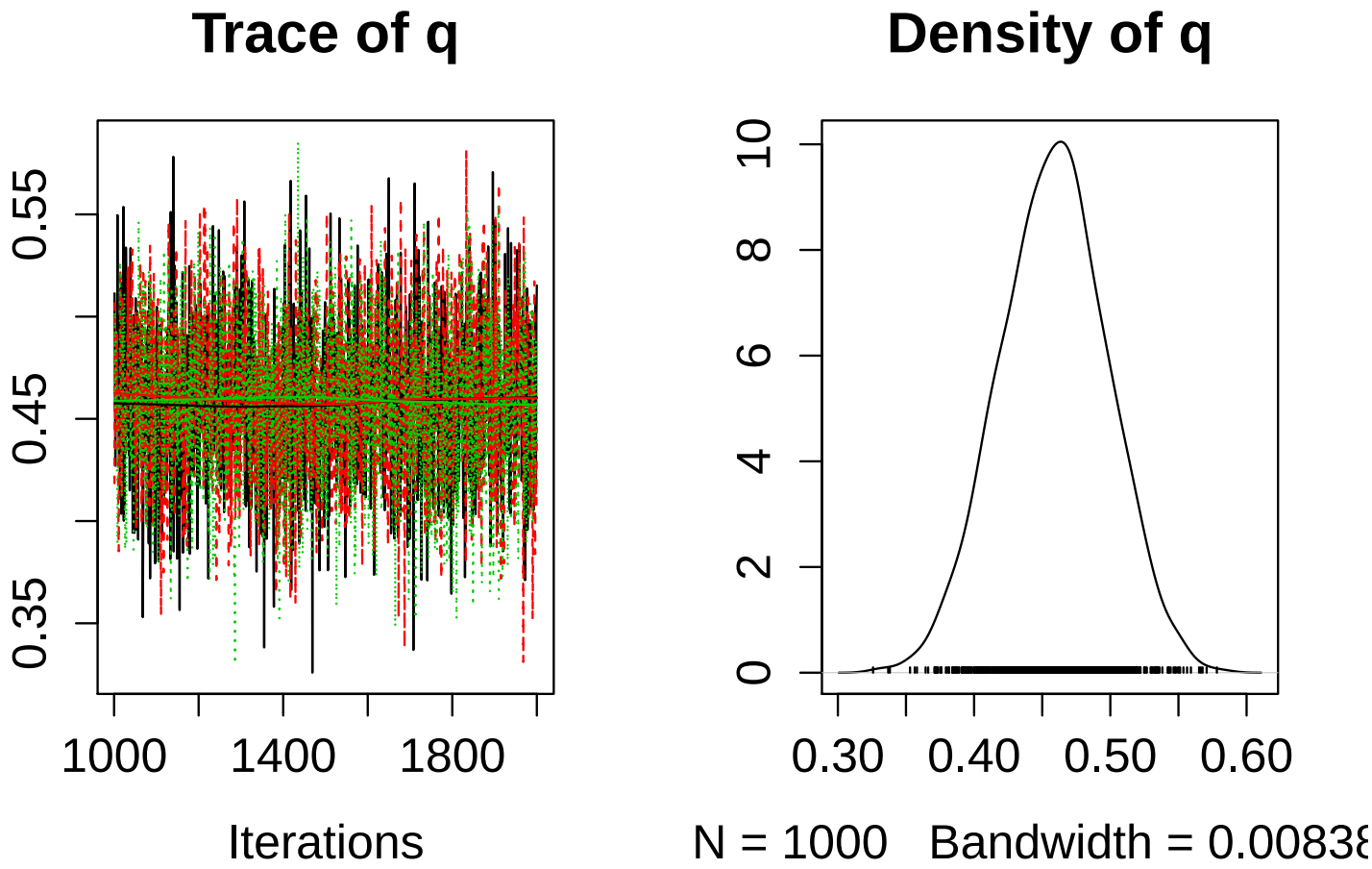
何やら
問題あり!

# 収束診断の $\hat{R}$ 指数

- `gelman.diag(post.mcmc.list)` → 実演表示
- `R-hat` は Gelman-Rubin の収束判定用の指数
  - $\hat{R} = \sqrt{\frac{\hat{\mathrm{var}}^{+}(\psi|y)}{W}}$
  - $\hat{\mathrm{var}}^{+}(\psi|y) = \frac{n-1}{n}W + \frac{1}{n}B$
  - $W$ : サンプル列内の variance の平均
  - $B$ : サンプル列間の variance
  - Gelman et al. 2004. Bayesian Data Analysis. Chapman & Hall/CRC

# Gibbs sampling → 事後分布の推定

- `plot(post.mcmc.list)`



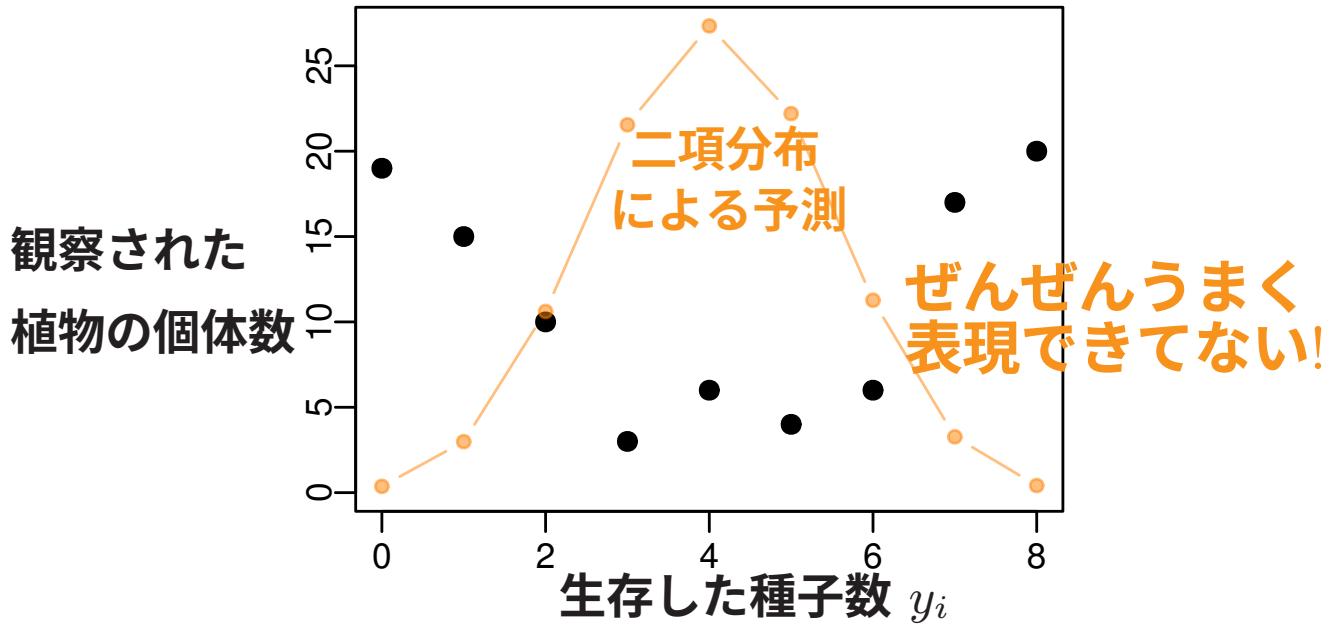
# 4. GLMM と階層ベイズモデル

## GLMM のベイズモデル化

hierarchical Bayesian
**階層ベイズモデルとなる**

# 二項分布では説明できない観測データ!

100 個体の植物の合計 800 種子中 403 個の生存が見られたので，平均生存確率は 0.50 と推定されたが……

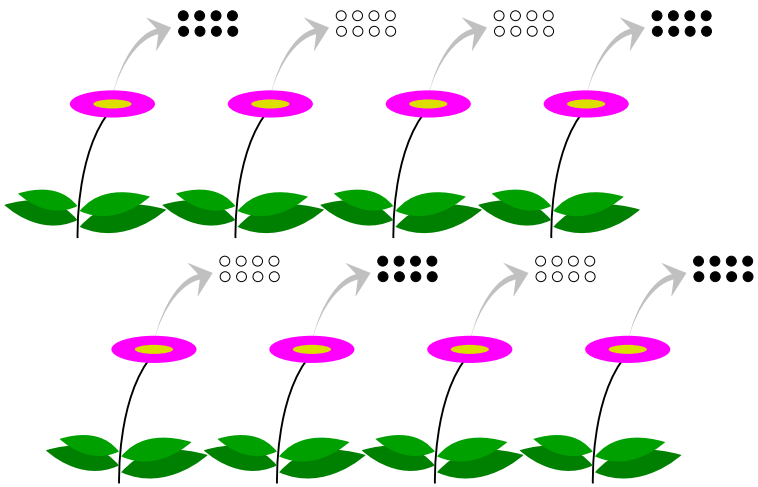


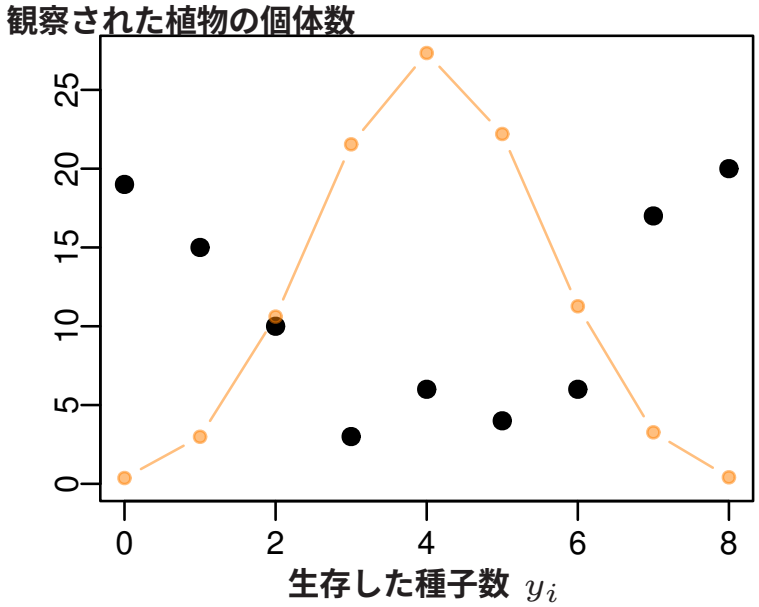

さっきの例題と同じようなデータなのに?

(「統計モデリング入門」第 10 章の最初の例題)

# 個体差 → 過分散 (overdispersion)

極端な過分散の例






- 種子全体の平均生存確率は 0.5 ぐらいかもしれないが……
- 植物個体ごとに種子の生存確率が異なる: 「個体差」
- 「個体差」があると overdispersion が生じる
- 「個体差」の原因は観測できない・観測されていない

# モデリングやりなおし: 個体差を考慮する

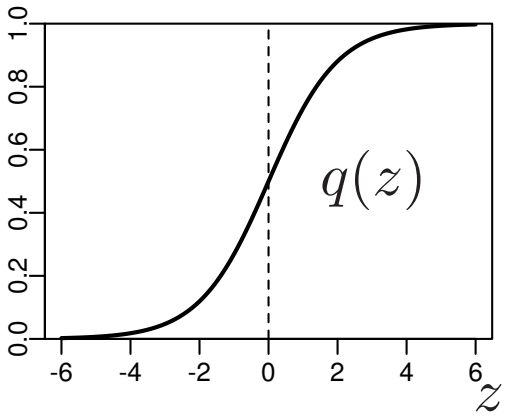
- 生存確率を推定するために 二項分布という確率分布を使う
- 個体 $i$ の $N_i$ 種子中 $y_i$ 個が生存する確率は二項分布

$$p(y_i \mid q_i) = \begin{pmatrix} N_i \\ y_i \end{pmatrix} q_i^{y_i} (1 - q_i)^{N_i - y_i},$$

- ここで仮定していること
  - 個体差があるので個体ごとに生存確率 $q_i$ が異なる

# GLM わざ: ロジスティック関数で表現する生存確率

- **生存確率** $q_i = q(z_i)$ **をロジスティック関数** $q(z) = 1/\{1 + \exp(-z)\}$ **で表現**




- **線形予測子** $z_i = a + r_i$ **とする**
  - **パラメーター** $a$**: 全体の平均**
  - **パラメーター** $r_i$**: 個体** $i$ **の個体差 (ずれ)**

# 個々の個体差 $r_i$ を最尤推定するのはまずい

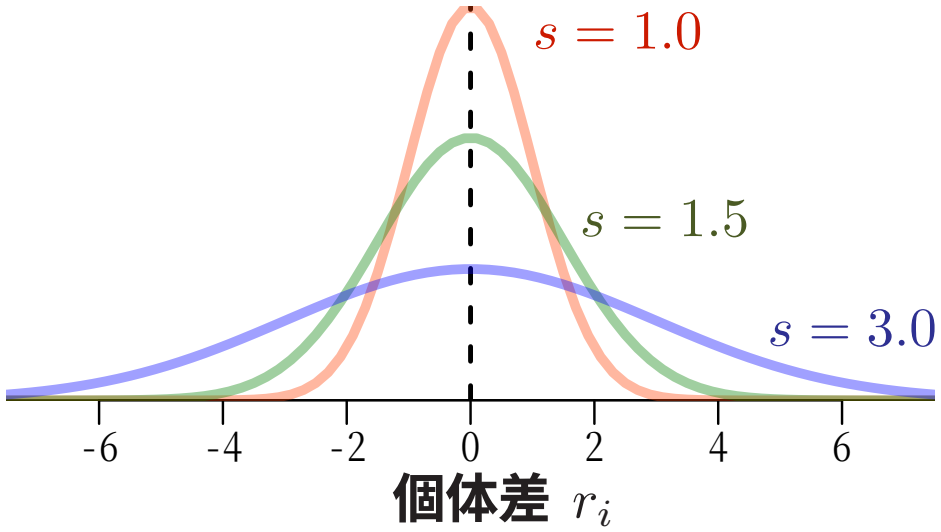
## パラメーター数 $>$ サンプルサイズ

- 100 個体の生存確率を推定するために**パラメーター 101 個** ($a$ と $\{r_1, r_2, \cdots, r_{100}\}$) を推定すると……
- 個体ごとに生存数 / 種子数を計算していることと同じ! (「データのよみあげ」と同じ)

## そこで，次のように考えてみる

suppose $\{r_i\}$ follow the Gausssian distribution

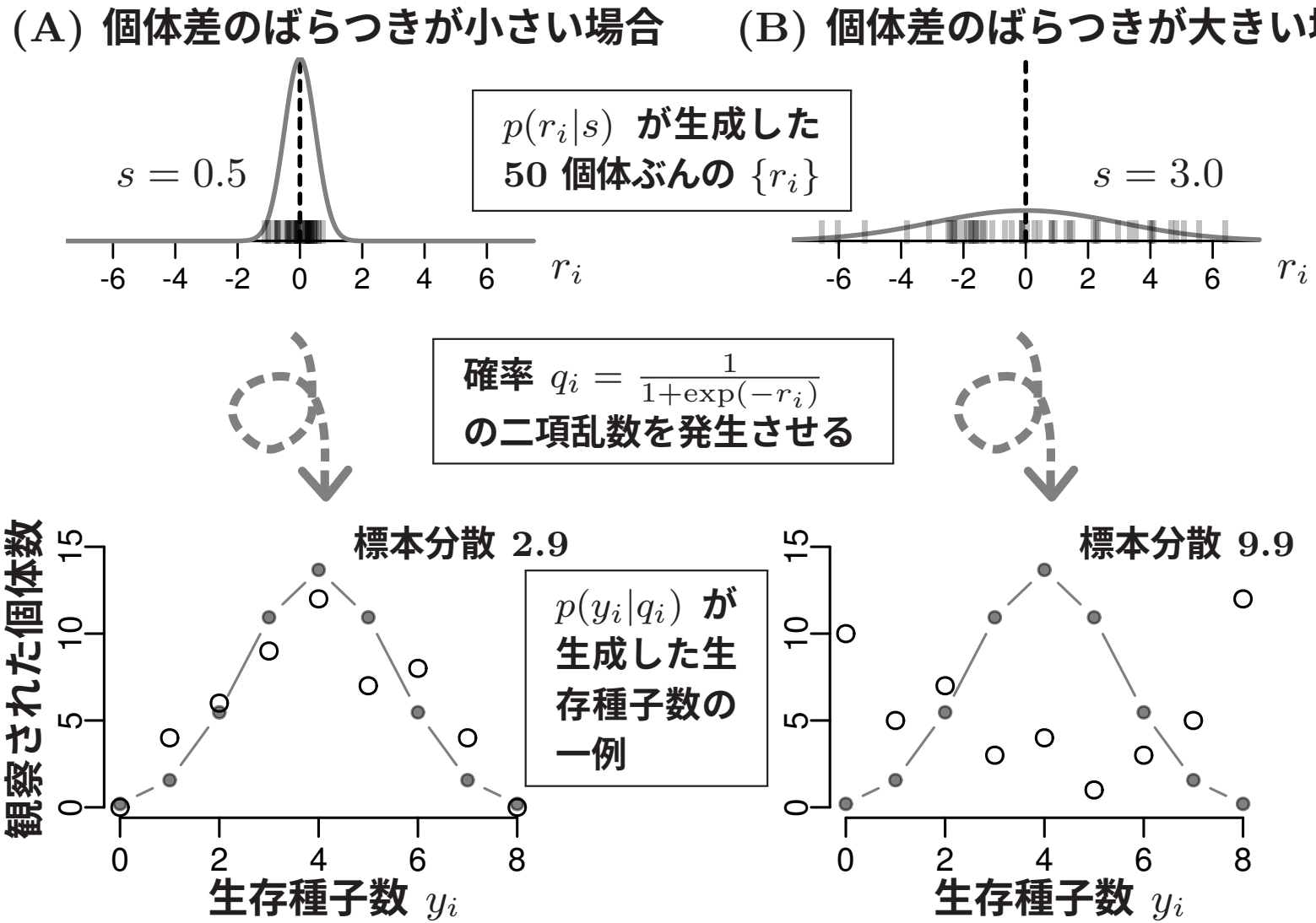
# $\{r_i\}$ のばらつきは正規分布だと考えてみる




$$p(r_i \mid s) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{r_i^2}{2s^2}\right)$$

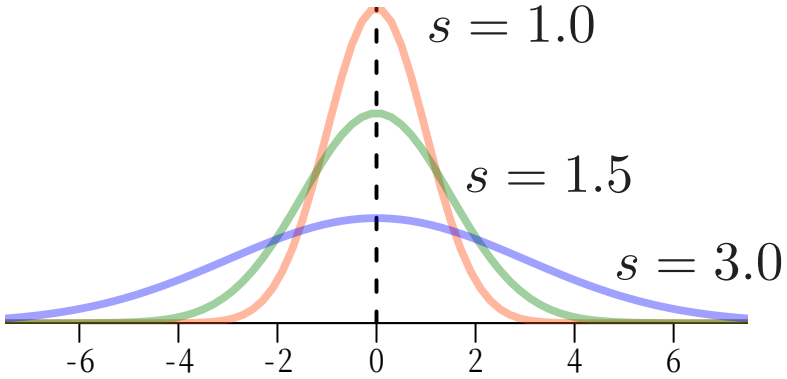
この確率密度 $p(r_i \mid s)$ は $r_i$ の「出現しやすさ」をあらわしていると解釈すればよいでしょう．$r_i$ がゼロにちかい個体はわりと「ありがち」で，$r_i$ の絶対値が大きな個体は相対的に「あまりいない」．

# ひとつの例示: 個体差 $r_i$ の分布と過分散の関係



# これは $r_i$ の事前分布の指定，ということ

前回の講義で $\{r_i\}$ は正規分布にしたがうと仮定したが
ベイズ統計モデリングでは 「100 個の $r_i$ たちに
共通する事前分布として正規分布 を指定した」
ということになる



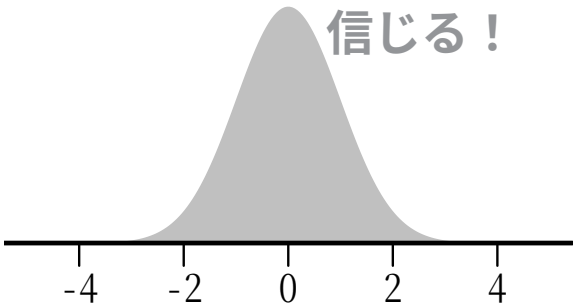

$$p(r_i \mid s) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{r_i^2}{2s^2}\right)$$

# ベイズ統計モデルでよく使われる三種類の事前分布

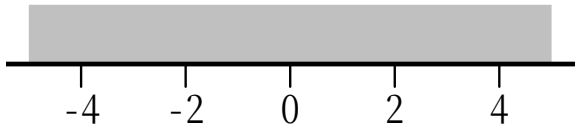
**たいていのベイズ統計モデルでは，ひとつのモデルの中で複数の種類の事前分布を混ぜて使用する．**

(A) **主観的な事前分布**

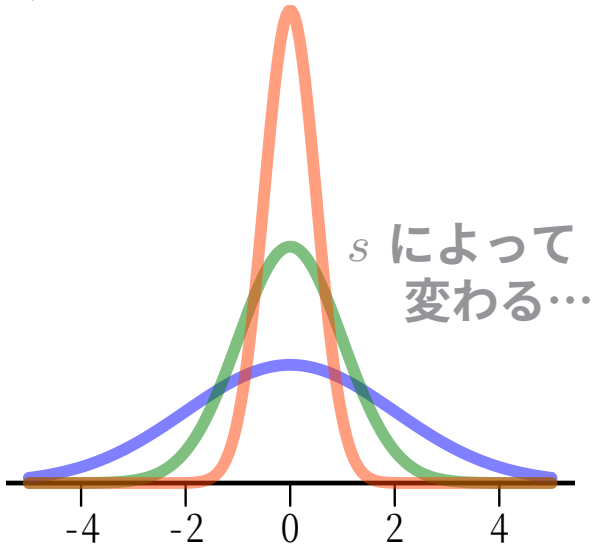
**(できれば使いたくない!)**




(B) **無情報事前分布**



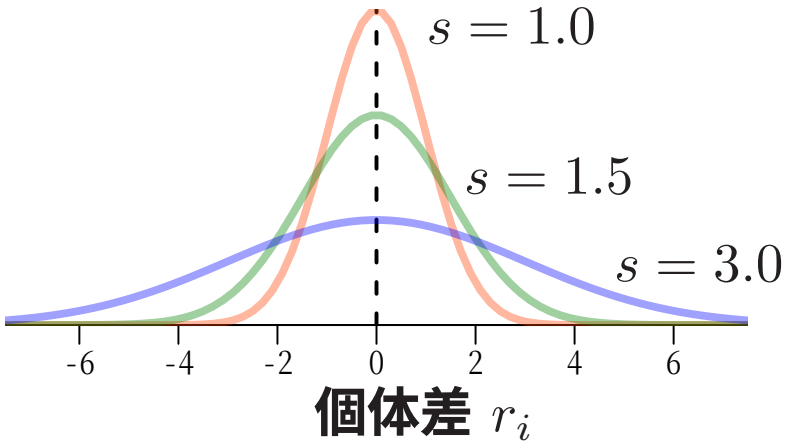

(C) **階層事前分布**



# $r_i$ の事前分布として階層事前分布を指定する

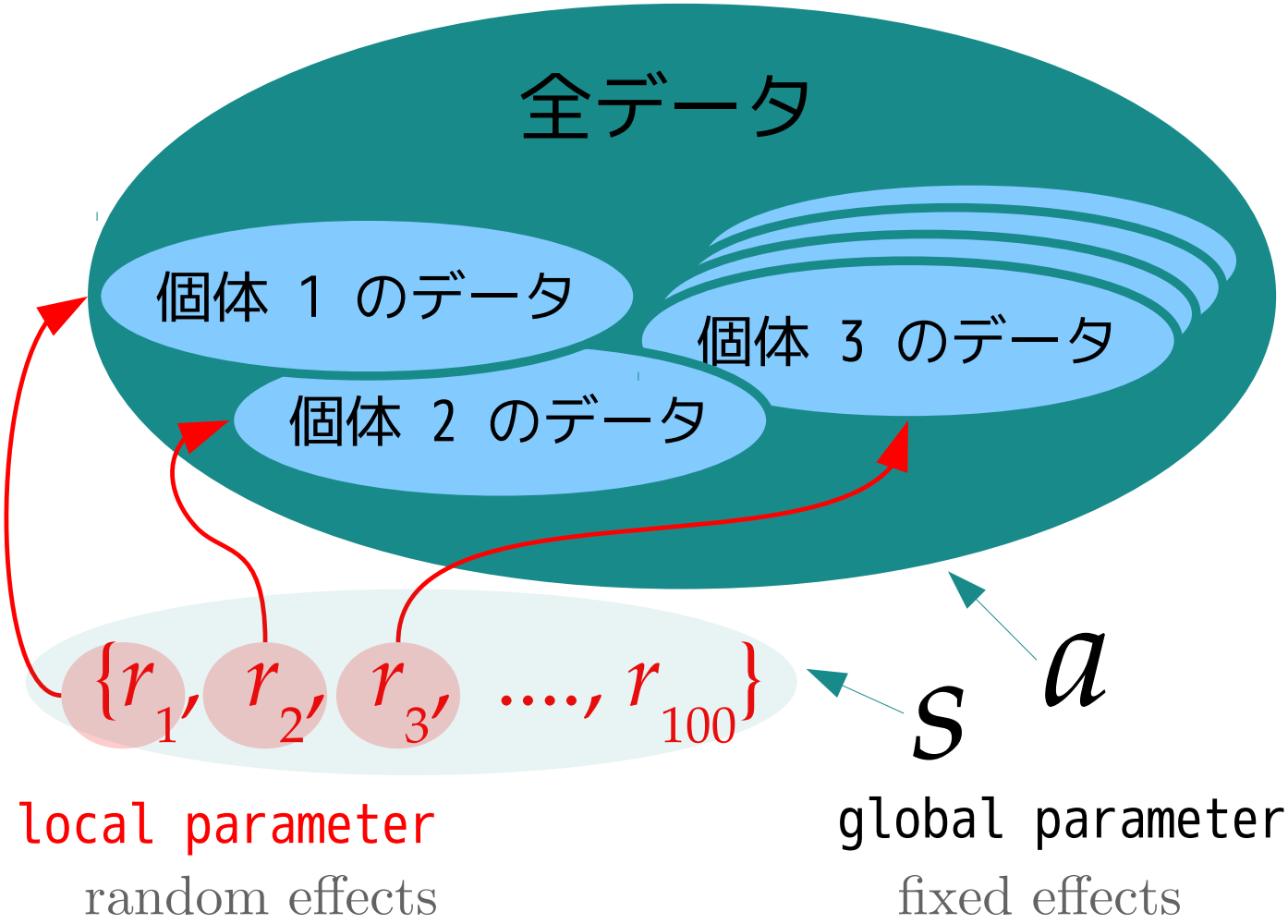
## 階層事前分布の利点

「データにあわせて」事前分布が変形!




$$p(r_i \mid s) = \frac{1}{\sqrt{2\pi s^2}} \exp\left(-\frac{r_i^2}{2s^2}\right)$$

# 統計モデルの大域的・局所的なパラメーター




**データのどの部分を説明しているのか?**

# パラメーターごとに適切な事前分布を選ぶ

(B) 無情報事前分布

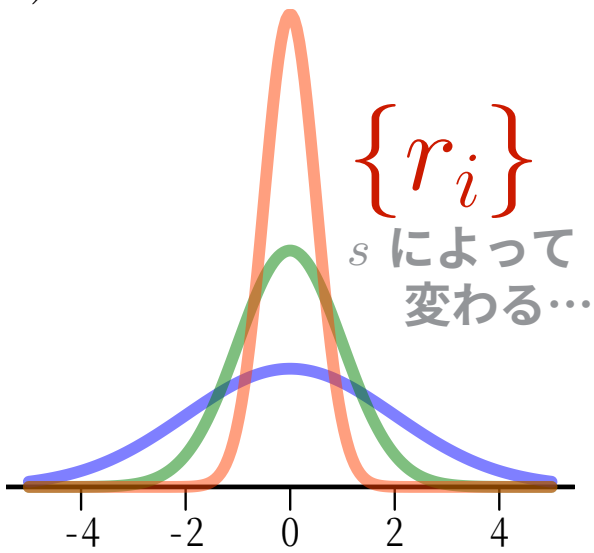
$a,\ s$

わからない？


(C) 階層事前分布

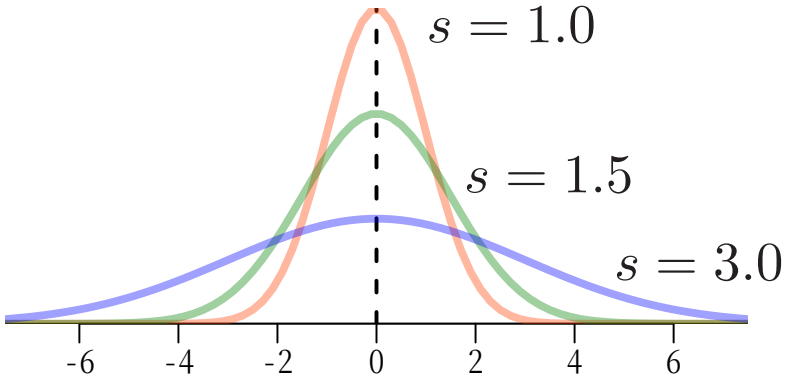
$\{r_i\}$

$s$ によって変わる…



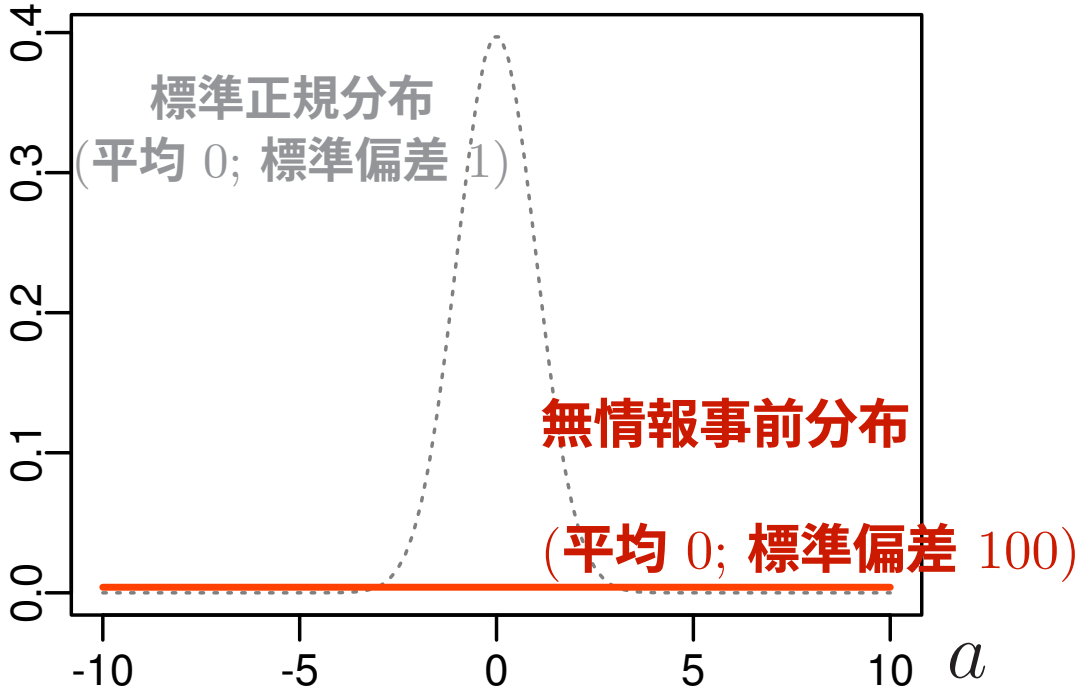

| パラメーターの種類 | 説明する範囲 | 事前分布 |
|---|---|---|
| 全体に共通する平均・ばらつき | global 大域的 | 無情報事前分布 |
| 個体・グループごとのずれ | local 局所的 | 階層事前分布 |

# 個体差 $\{r_i\}$ のばらつき $s$ の無情報事前分布




- $s$ はどのような値をとってもかまわない
- そこで $s$ の事前分布は **無情報事前分布** (non-informative prior) とする
- たとえば一様分布，ここでは $0 < s < 10^4$ の一様分布としてみる

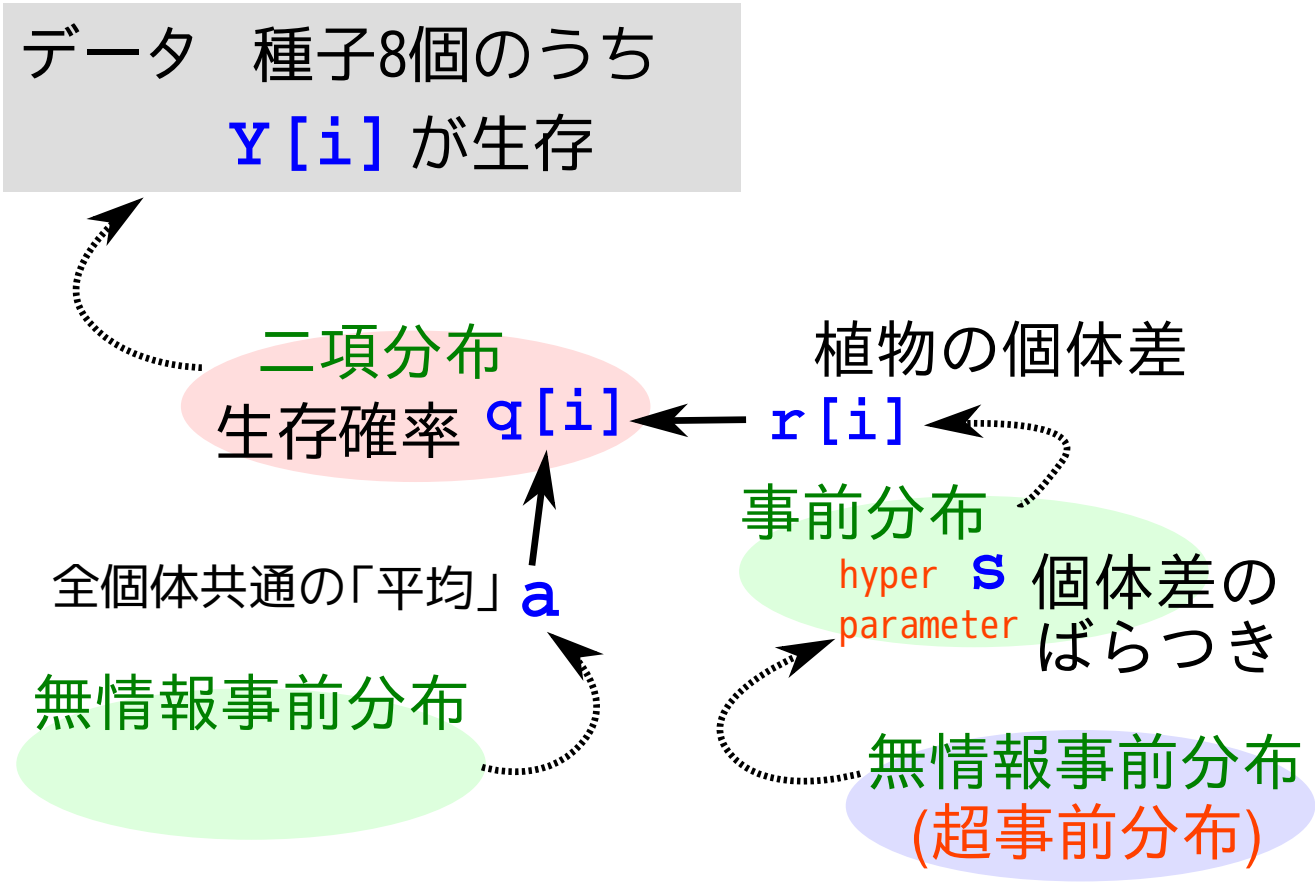
# 全個体の「切片」 $a$ の無情報事前分布




「生存確率の (logit) 平均 $a$ は何でもよい」と表現している

# 階層ベイズモデル: 事前分布の階層性

**超事前分布 → 事前分布という階層があるから**




**矢印は手順ではなく，依存関係をあらわしている**

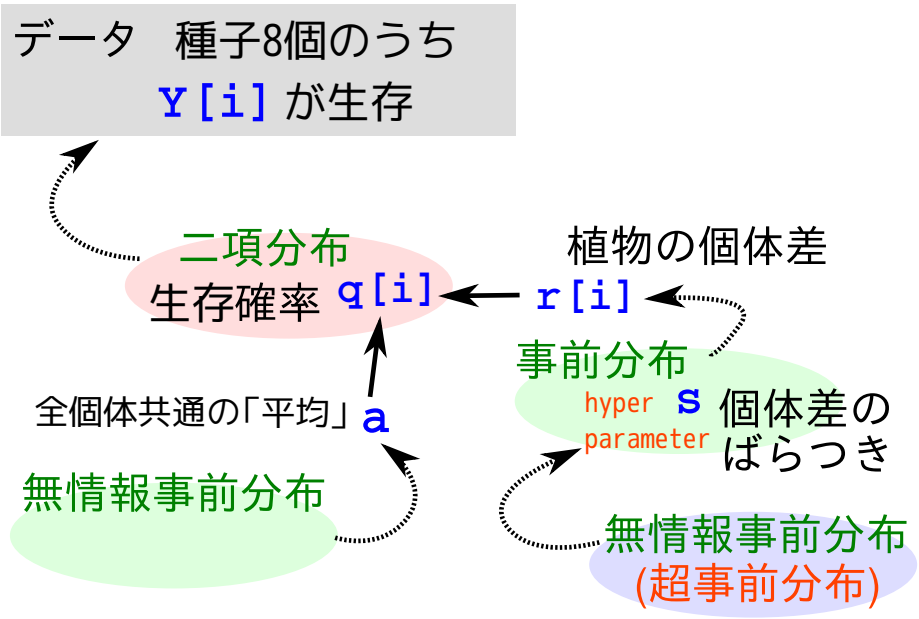
# 5. 階層ベイズモデルの estimation 推定

## ソフトウェア JAGS を使ってみる

R の “したうけ” として JAGS を使う

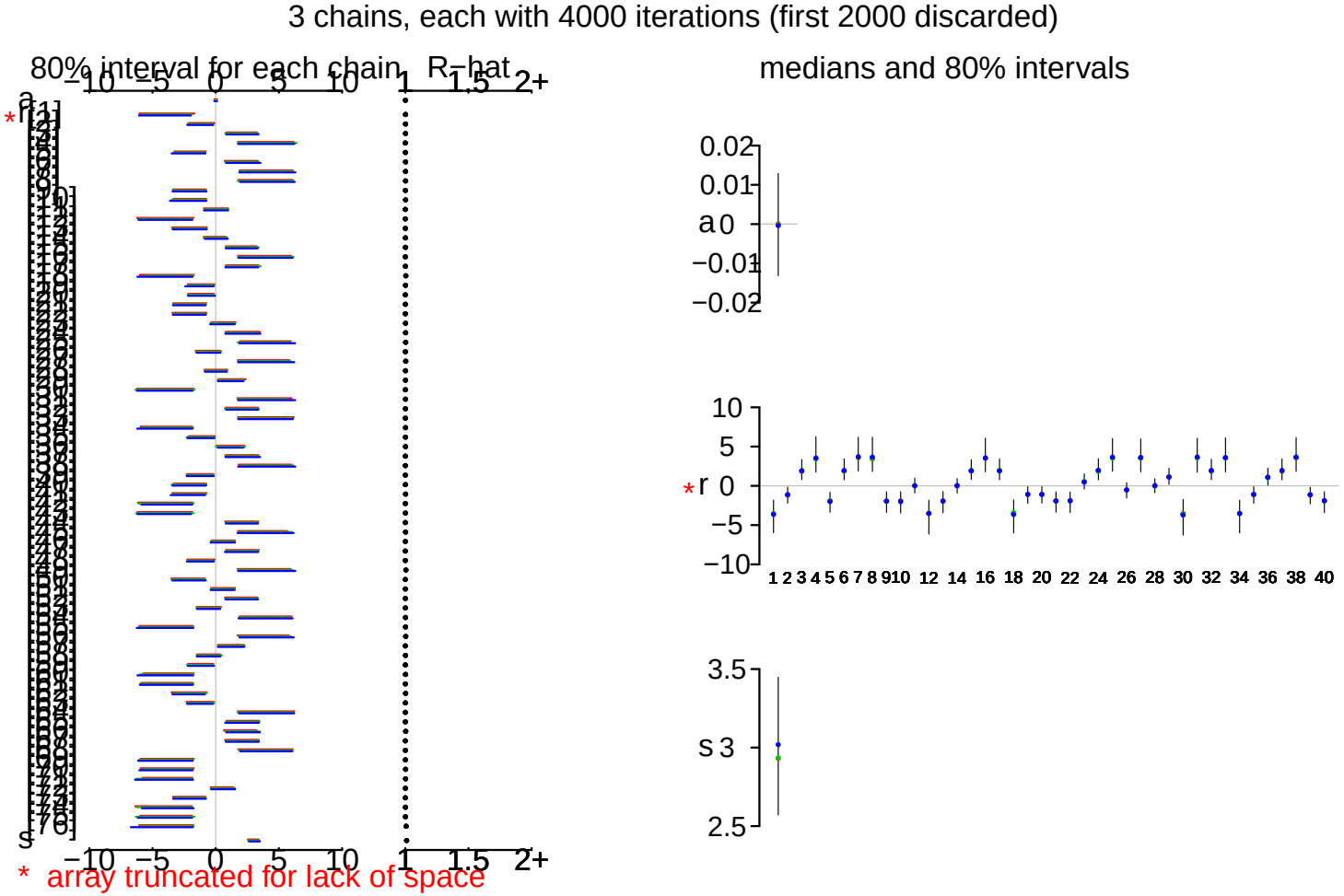
# 階層ベイズモデルを BUGS コードで記述する

```
model
{
  for (i in 1:N.data) {
    Y[i] ~ dbin(q[i], 8)
    logit(q[i]) <- a + r[i]
  }
  a ~ dnorm(0, 1.0E-4)
  for (i in 1:N.data) {
    r[i] ~ dnorm(0, tau)
  }
  tau <- 1 / (s * s)
  s ~ dunif(0, 1.0E+4)
}
```



# JAGS で得られた事後分布サンプルの要約

```
> source("mcmc.list2bugs.R")                    # なんとなく便利なので…
> post.bugs <- mcmc.list2bugs(post.mcmc.list) # bugs クラスに変換
```

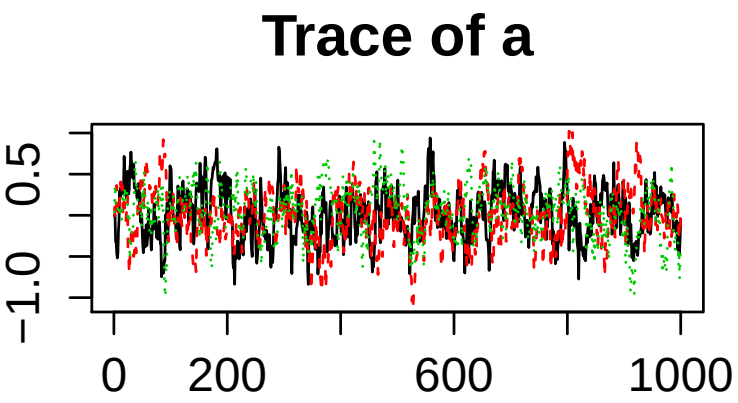


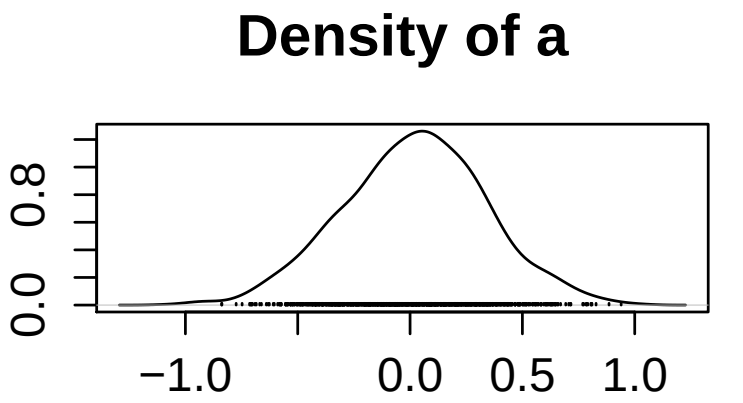
# bugs オブジェクトの post.bugs を調べる

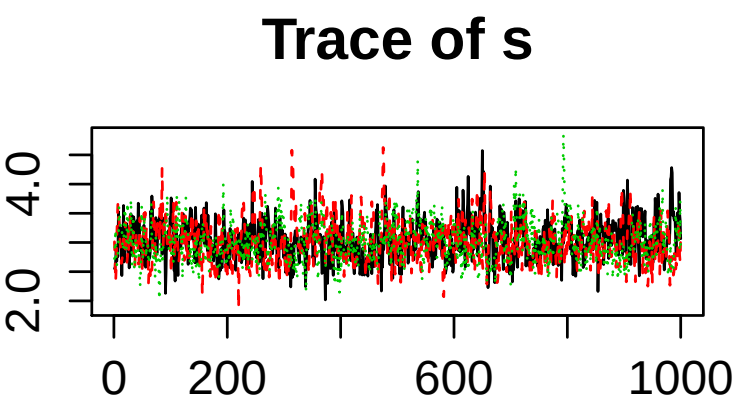
- print(post.bugs, digits.summary = 3)
- **事後分布の 95% 信頼区間などが表示される**

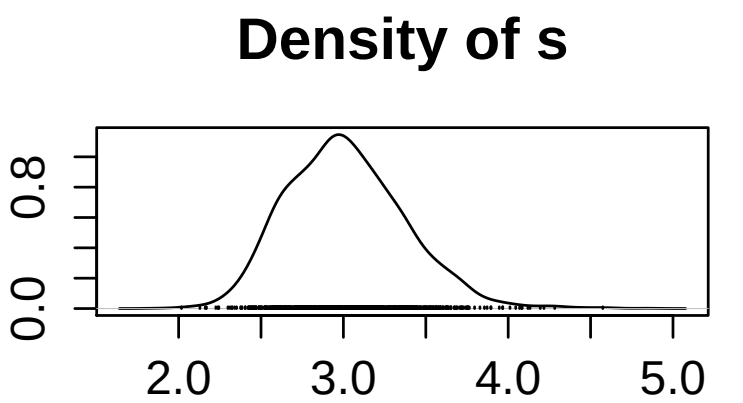
```
 3 chains, each with 4000 iterations (first 2000 discarded), n.thin = 2
 n.sims = 3000 iterations saved
         mean    sd   2.5%    25%    50%    75%  97.5%  Rhat n.eff
a       0.020 0.321 -0.618 -0.190  0.028  0.236  0.651 1.007   380
s       3.015 0.359  2.406  2.757  2.990  3.235  3.749 1.002  1200
r[1]   -3.778 1.713 -7.619 -4.763 -3.524 -2.568 -1.062 1.001  3000
r[2]   -1.147 0.885 -2.997 -1.700 -1.118 -0.531  0.464 1.001  3000
r[3]    2.014 1.074  0.203  1.282  1.923  2.648  4.410 1.001  3000
r[4]    3.765 1.722  0.998  2.533  3.558  4.840  7.592 1.001  3000
r[5]   -2.108 1.111 -4.480 -2.775 -2.047 -1.342 -0.164 1.001  2300
... (中略)
r[99]   2.054 1.103  0.184  1.270  1.996  2.716  4.414 1.001  3000
r[100] -3.828 1.766 -7.993 -4.829 -3.544 -2.588 -1.082 1.002  1100
```

# 各パラメーターの事後分布サンプルを R で調べる



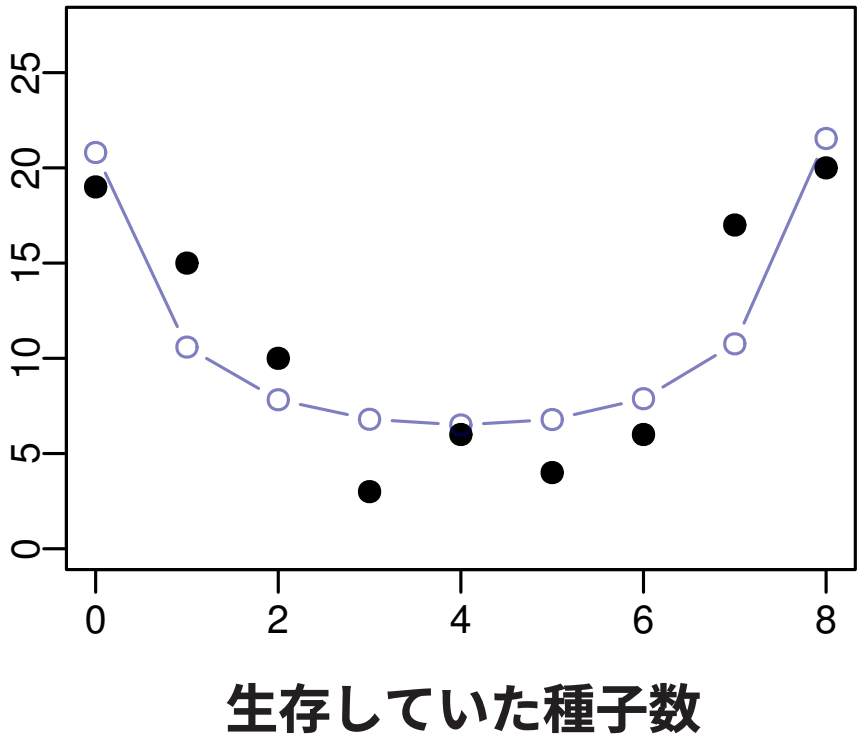



# 得られた事後分布サンプルを組みあわせて予測

- `post.mcmc <- to.mcmc(post.bugs)`
- **これは `matrix` と同じようにあつかえるので，作図に便利**
- **……このあとごちゃごちゃと計算する必要あるけど，省略……**



# 次回予告

The next topic

# 階層ベイズモデルと時間変化モデル

Hierarchical Bayesiam Model (HBM) & Time Change Model